Oyster 백서 개정본 버전 0.7b

작성일자: 2017년 9월

작성자: Bruno Block

bruno@oyster.ws

oysterprotocol.com

Oyster Protocol은 웹사이트들이 사이트 방문자들을 성가시게 하지 않으면서 수익을 올릴 수 있게 하는데 그 이유는 이 방문자들이 분산되어 있는 원장(ledger)에 대해 작업 증명을 수행하도록 하기 때문이다.

개요

인터넷의 기하급수적 성장에도 불구하고 웹 컨텐츠를 금전적 수익으로 전환시키는 방식들은 여전히 정체 상태에 머물러 있어왔다. 광고는 사생활을 침해하고, 사용자가 원하는 컨텐츠에 집중할 수 없게 만들고, 웹사이트들의 디자인 일관성을 깨뜨린다. 일반적으로 사람들이 온라인 광고를 무시하고 이에 대해 부정적 정서를 갖고 있기 때문에 이들 대다수가 광고 차단 프로그램들을 사용하게 됐다. 컨텐츠 공급자들은 광고 차단 프로그램을 감지하면 해당 이용자들을 차단하거나 제한함으로써 이들을 배척하는데 까지 이르게 됐다. 컨텐츠 공급자들은 광고 차단 프로그램들로 인해 상당한 금전적 손실을 보고 있거나 광고 차단 보복 메커니즘 적용으로 인해 상당수의 이용자들을 잃고 있다. 따라서 광고계 전체가 점진적으로 미래에 대한 예측이 반영된 통합적 해결책을 적용하지 못한 채 비효율적이고 비효과적이며 사생활 침해 요소가 큰 괴로운 경험을 제공하는 장으로 변형되었다.

더불어 현재 시점에서 편리하면서 동시에 사생활을 보장해주는 저장 서비스는 존재하지 않는다. 편의성을 추구한다면 일반적인 클라우드 저장 회사를 선택하게 되는데 이들은 사생활과 익명성을 제공하지 않는다. 이들이 소스 비공개 소프트웨어를 사용한다는 것 사실 자체가 이들이 말하는 것을 사용자가 절대 고지 곳대로 믿을 수 없다는 뜻이다. 반대로 사생활을 선택할 경우 간단하게 '올리기' 버튼만 갖춰져 있는 단순한 웹 인터페이스를 갖춘 저장 서비스를 찾기란 거의 불가능하다.

Oyster Protocol은 진정으로 일석이조를 가능하게 하는 방식이다. Oyster Protocol은 컨텐츠 공급자들과 소비자들이 균형과 협력을 할 수 있도록 하는 완전히 다른 접근 방식을 소개한다. 이에 따라 웹 브라우저만 있으면 누구든 파일들을 분산시켜서, 익명으로, 안전하게, 안심하고 저장했다가 다시 가져 와서 사용하는 것이 가능하다.

다음은 Oyster 생태계 구성 주체들이다:

저장 공간 사용자(Storage User) - Oyster Pearl을 지불하고 파일을 업로드 하는 사용자

<u>책임</u>

- 정확한 액수의 Oyster Pearl을 두 개의 중개 노드(Broker Node)에 지불한다.
- 중개 노드 선택은 자동으로 이뤄지게 되어 있긴 하지만 사용할 두 개의 중개 노드를 어떤 것으로 할 것인지에 대해 최종 결정은 저장 공간 사용자가 한다.
- 브라우저에서 파일을 암호화 하고 분할한 후 이들은 앞서 선택한 중개 노들들에게 전송한다.
- 중개 노드들이 설치한 데이터 지도(Data Map)의 무결성을 검증한다.
- 분산 평판 시스템(Distributed Reputation System)을 통해 중개 노드 계약들을 공유한다.
- 이후 Tangle 로부터 파일을 다시 불러올 때 필요한 Oyster Handle을 안전하게 저장한다.

<u>보상</u>

- 파일이 안전하고, 믿을 수 있으며, 익명으로 저장된다.

웹사이트 소유자(Website Owner) – 웹사이트를 운영하는 조직이나 개인

<u>책임</u>

- 웹 노드들(Web Node)에게 컨텐츠/재화/용역을 제공한다.
- 자신의 웹사이트 HTML에 Oyster Protocol 스크립트를 추가한다.

보상

- 웹 노드들이 발견한 Oyster Pearl을 지급받는다.

웹 노드(Web Node) – 웹사이트에 접속하는 웹 브라우저

책임

- 작업 증명을 통해 보물 지도들(Treasure Maps)을 뒤져서 묻혀 있는 Oyster Pearl들을 찾아낸다.
- 찾은 보물을 중개 노드에 제출하고 웹 사이트 소유자를 대신하여 결제를 청구한다.
- 중개 노드들을 위해 작업 검증을 수행함으로써 이들이 웹 노드 신원과 보물 지도를 확보할 수 있도록 한다.
- 웹 노드들을 위해 작업 검증을 수행함으로써 이들이 웹 노드 신원과 보물 지도를 확보할 수 있도록 한다.
- 알맞게 작업 검증을 수행한 웹 노드들에게 웹 노드 신원과 이전 보물 지도를 전송한다.
- 분산 평판 시스템을 통해 중개 노드 계약들을 공유한다.

보상

- 접속하는 웹 사이트의 소유자로부터 컨텐츠/재화/용역에 대한 접근을 허가 받는다.
- 경우에 따라 작업 증명 부담을 다른 웹 노드들에게 넘겨주기도 한다.

중개 노드(Broker Node) - Tangle과 블록체인(Blockchain)에 접속하는 네트워크 장비

책임

- 상호 간에 인접해 있는 노드들을 통해 Tangle 에 대한 접속을 유지한다.
- 웹 노드들과 저장 공간 사용자들이 Tangle에 접속할 수 있도록 한다.
- 새로운 파일이 업로드될 경우 작업 증명을 수행한다.
- 저장 공간 사용자의 Oyster Pearl을 블록체인 계약 내에 있는 묻혀 있는 상태로 제출한다.
- 작업 증명이 올바르게 수행됨으로써 발견된 보물의 잠금 상태를 해제한다.
- 발견된 보물의 잠금 상태를 해제할 수 있도록 이더리움 잔액을 플러스 상태로 유지한다.
- 분산 평판 시스템 상에 평판 점수를 구축한다.
- 웹 노드들 간에 peer-to-peer 연결 개시 작업을 중개한다.
- 작업 증명을 수행하는 웹 노드들에게 새로운 보물 지도들을 전송한다.

보상

- 새로 묻히고 남은 보물들을 수집함으로써 Oyster Pearl을 얻는다.
- 새로 발견된 보물로부터 발생한 수수료를 징수하여 Oyster Pearl을 얻는다.
- 경우에 따라 작업 증명 부담을 웹 노드들에게 넘겨주기도 한다.

IOTA Tangle – 분산되어 있는 원장으로서 비순환 방향 그래프(Directed Acyclic Graph,

DAG)라고 불린다

책임

- 작업 증명이 수행된 데이터를 유지한다
- 중복된 데이터 복사본들을 지리적으로 분산시킨다.
- Swarm Intelligence 등을 이용하여 저장 부하를 분산시킨다.

보상

- 네트워크 상에 공격 벡터에 대한 저항력이 향상된다.
- 거래 확인에 소요되는 평균 시간이 빨라진다.

이더리움 블록체인**(Ethereum Blockchain)** – 스마트 계약 기능을 갖춘 분산 원장

책임

- Oyster Pearl에 내재되어 있는 속성들을 (토큰으로) 생성하기 위한 스마트 계약 틀(Smart Contract Framework)을 제공한다

보상

- 블록체인 채굴자들이 중개 노드들로부터 수수료를 이더리움으로 지급받는다.

Tangle의 구조

IOTA Tangle 은 비순환 방향 그래프(DAG)이다. 즉 블록을 형성하지 않는 분산 원장이라는 뜻이다. IOTA Tangle을 실시간으로 시각적으로 도식화한 모습은 여기에서 확인이 가능하다. 제출된 각각의 거래는 그 이전에 발생한 두 개의 거래들에 대해 작업 검증을 수행해야만 하는 데 이로써 이들 두 거래를 확정시키게 된다. 이 두 가지 거래들을 문맥상 가지와 몸통이라 칭한다. 거래를 Tangle로 전송하는 것에 관한 보다 상세한 정보는 여기를 참조하면 된다. 각 거래들은 페이로드란 것이 있는데 이것을 이용하여 저장 공간 사용자가 업로드한 데이터를 유지한다. 거래들은 상호 peer to peer로 연결되어 있는 노드들로 이뤄진 메쉬 네트워크 전체로 전파되는 한편 각 노드들은 이러한 거래들의 중복된 복사본을 유지한다. 이로 인해 상당한 규모의 데이터 사본들의 중복성을 일으키고 이에 따라 중압집중화된 호스팅 공급자에 의존함이 없이 데이터 손실의 위험을 대폭 경감시킬 수 있게 된다.

Tangle 노드들은 일단 자신들의 물리적인 저장 한계가 꽉 차게 되면 이전 데이터들을 자동으로 삭제하도록 설계되어 있다(이를 Automatic Snapshot이라 부르는데 아직 Tangle 상에 서 실제 발생하고 있지는 않다). 이것은 결국 거래 데이터가 노드에서 삭제되게 된다는 것을 의미한다. 따라서 웹 노드들은 묻혀 있는 Oyster Pearl을 찾는 과정에서 작업 증명을 수행 하여 거래 데이터를 Tangle에 다시 첨부하게 된다. 이로써 Tangle 노드들 전체에 걸쳐 데이터가 유지되도록 하고 복구 불가능한 상태로 삭제가 되지 않도록 한다.

IOTA Tangle은 그 개발 계획 상에 다수의 혁신들이 포함되어 있는데 그 중에서 Swarm Intelligence를 손꼽을 수 있다. Swarm Intelligence는 전체 원장을 유지하는 데 필요한 각 Tangle에 발생하는 병목 현상을 제거해주기 때문에 Oyster Protocol에 중요하다. 그것은 마치 드라이브 배열 설정을 RAID 1에서 RAID 10으로 전환하는 것과 같다고 할 수 있다. Swarm Intelligence의 구현은 Oyster Protocol의 확장성 측면의 장점을 더욱 강화시킨다.

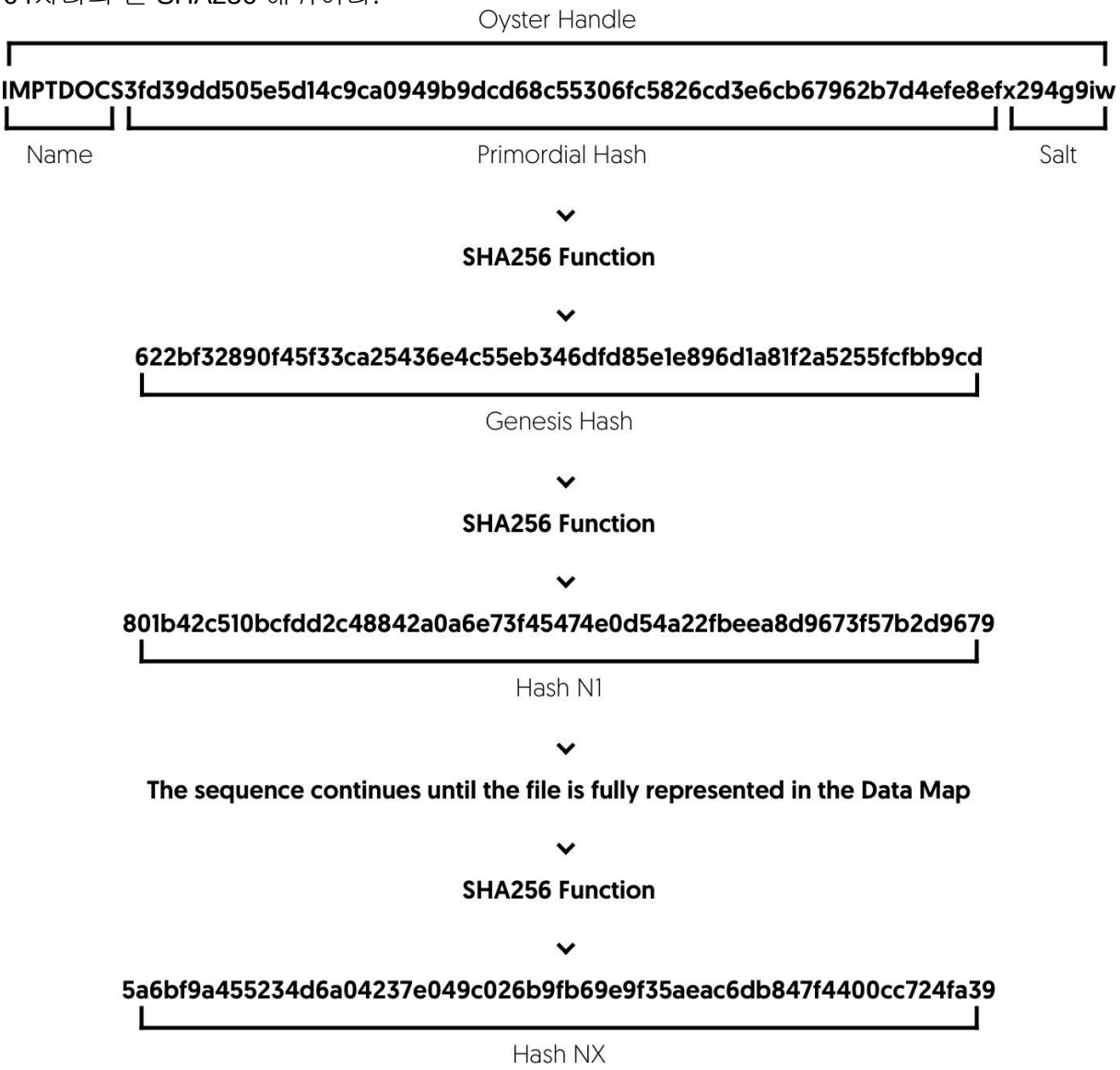
네트워크 대역폭은 Tangle 상에서 데이터를 믿고 전송할 수 있도록 하는 측면에 있어서 가장 희귀한 자원이다. Tangle 노드는 내제적으로 네트워크 인터페이스 대역폭에 의해 제약을 받게 되어 있다. Oyster Protocol은 Tangle로의 데이터 전송을 보다 안심하고 수행할 수 있도록 하는 것을 목표로 하기 때문에 Tangle에 대한 대역폭 접속에 대해 처리된 Oyster Pearl 들 중 일부분을 보상으로 제공한다. Oyster Protocol 사양을 충족하는 Tangle 노드들을 중개 노드라 칭한다. 중개 노드들은 Tangle을 웹 노드 및 저장 공간 사용자들과 연결시키는 교량 역할을 한다.

데이터는 최대 크기가 1KB 미만인 부분 여러 개로 나뉘어져서 거래 페이로드 내에 저장된다. SHA256 해쉬는 Tangle 상에서 데이터를 저장하고 불러오는 데 사용되는 참조 기준이다. SHA256 해쉬로 데이터를 대표하기로 할 경우 해당 데이터는 삼진법 형태로 변환되어 이것이 해당 데이터의 수신자 주소를 대표하게 된다. 해당 데이터를 Tangle에서 다시 불러오고자 할 경우 해당 해쉬가 다시 삼진법 형태로 변환되어 수신자 주소를 생성하고, 그 다음 해당 주소 아래 있는 모든 거래 데이터가 복구된다. 이렇게 선택한 해쉬를 대표하는 페이로드 데이터를 담고 있는 거래는 발행 시점이 가장 오래된 거래이다.

Tangle 상에 파일 최초 저장

저장 공간 사용자가 Oyster Protocol을 통해 파일을 업로드 하고자 할 경우 해당 파일은 브라우저 상에서 여러 부분으로 분할되고 암호화된다. 이렇게 파일을 여러 부분으로 분할해서 고립시키으로써 악의적 이용자가 해당 데이터를 확보하는 것을 불가능하게 하는데 그 이유는 이 데이터에 접속하기 위해서는 Oyster Handle 이라는 암호화 키가 있어야만 가능하기 때문이다.

Oyster Handle 의 첫 8자리는 해당 파일의 이름을 나타낸다. 이 이름은 통상 브라우저로 업로드된 파일의 이름을 복사해서 생성되지만 저장 공간 사용자가 원하는 데로 수정할 수 있다. Primordial Hash는 저장 공간 사용자의 브라우저 내에서 무작위성을 최대화하여 임으로 생성된 64자리의 긴 SHA256 해쉬이다.



Oyster Handle의 끝 8자리는 cryptographic salt 로서 Primordial Hash를 암호화 키 전체와 구분하는 역할을 한다. 이 Salt라는 것은 미래에 해쉬 함수 상에 취약점이 생기거나 Genesis Hash 상에 rainbow table 공격으로 인해 Primordial Hash 가 발견될 경우에 대비해 데이터 보호를 보강하는 데 사용된다. 따라서 80자리로 구성된 Oyster Handle은 여러 부분으로 분할된 데이터를 암호화하고 복호화하는 데 사용되는 전체 암호화 키가 되는 것이다. Oyster

Protocol은 또한 암호에 암호문구를 추가할 수 있도록 한다. Primordial Hash가 분할된 데이터 부분들을 대표할 SHA256 해쉬 생성 절차를 시작한다. 가장 먼저 데이터가 최대 1KB 크기의 부분 여러 개로 분할되고, 이후 각각의 부분이 개별적으로 암호화되면서 전체 Oyster Handle이 암호화 키로서 생성된다. 분할된 각 데이터 부분이 순차적으로 반복 생성되어 해쉬(Genesis, N1, N2 등)로 표현되고 결국 이들 각각이 두 개의 중개 노드들에 의해 하나의 Tangle 거래로서 제출된다.

분산평판시스템이 웹 노드들과 저장 공간 사용자들 사이에서 가장 성과가 좋은 중개 노드들을 추적하고, 그에 따라 저장 공간 사용자를 대신하여 가장 적합한 두 개의 중개 노드들을 자동으로 선택한다. Oyster Protocol에서는 정확히 두 개의 중개 노드들을 선택하도록 명시하고 있는 데 이는 이 둘 사이에 경쟁을 유발하여, 이 경쟁에 이기기 위해 중개 노드는 대부분의 Oyster Pearl을 데이터 지도에 파묻게 된다. 바로 이 이유로 인해 성과가 더 좋은 중개 노드가 Oyster Pearl을 덜 받게 되긴 하지만 대신 더 좋은 평판을 얻게 되어 미래에 Oyster Pearl을 얻을 수 있게 되는 것이다.

Genesis Hash 값이 저장 공간 사용자에 의해 두 개의 중개 노드들에게 제출된다. 이 노드들 중 하나는 데이터 지도를 Genesis Hash 로부터 하향으로 확정하는 역할로 지정하고(알파 노드) 다른 노드는 데이터 지도를 NX Hash(X는 해쉬 반복 생성 작업 중 마지막 회차를 나타낸다) 로부터 상향으로 확정하는 역할로 지정한다(베타 노드). 알맞은 금액의 Oyster Pearl이 알파 노드로 지정된 중개 노드로 전송된다. 알파 노트로 Pearl 금액 전액과 베타 노드로 지정된 중개 노드의 이더리움 주소가 전송된다. 알파 노드는 자신의 신분을 증명하는 암호 형태로 서명된 증명서와 함께 Pearl을 받으면 그 절반을 베타 노드에 전송한다. 이 두 노드들 중 어느 하나라도 이탈을 할 경우 이는 분산 평판 시스템을 통해 보고되고 이렇게 되면 이들의 평판은 Oyster 네트워크 전체에 있는 웹 노드들과 저장 공간 사용자들 사이에 엄청난 손상을 입게 된다. 저장 공간 사용자가 지불하는 Pearl 금액은 나중에 데이터 지도에 묻히게 되는 금액의 절반이다. 중개 노드들은 데이터 지도에 묻히고 남은 Pearl을 자신들이 가질 수 있다.

작업 증명을 수행하여 각 분할된 데이터 각 부분들을 Tangle에 올리는 작업은 선택된 중개 노드들이 수행하도록 기본 설정이 되어 있다. 거래를 전송하는 데 쓰이는 Tangle 주소는 반복 작업을 통해 순차적으로 생성된 해쉬들(Genesis, N1, N2 등) 중 해당되는 해쉬값이 삼진법 형태로 표시된 것이다. 그러나 peer-to-peer 연결 중개와 새로운 Genesis Hash에 대한 수요가 충분히 있을 경우 중개 노드들은 작업 검증 작업을 웹 노드들에게 위임할 수 있다.

Tangle 네트워크 전체에서 이뤄지는 전체 작업 검증 작업이 증가할 경우 이것이 거래 확정에 소요되는 시간을 줄이고 Tangle 네트워크 전반의 안정성을 향상시키는 데 이러한 현상이 발생하는 이유에 대해 보다 상세히 알고 싶을 경우 이곳을 참조하면 된다.

중개 노드로 **Pearl** 파묻기

Oyster Pearl은 업로드된 파일의 구조와 컨텐츠를 정의해둔 데이터 지도 내부에 묻히도록 설계되어 있다. 저장 공간 사용자들이 직접 Pearl을 데이터 지도에 묻도록 하지 않고 대신 중개 노드들에게 이 임무를 배정하는 데는 다음과 같은 여러 이유들이 있다.

- 중개 노드들이 이더리움 구좌에 접속할 수 있다. 따라서 이들은 모든 Pearl을 지정된 곳들로 정확히 이동시키는 데 필요한 가스를 지불할 수 있다. 임의의 크기를 가진 사용자 데이터들 각각에 대해 이더리움 블록체인 상에서 반드시 거래가 수행되어야 한다.

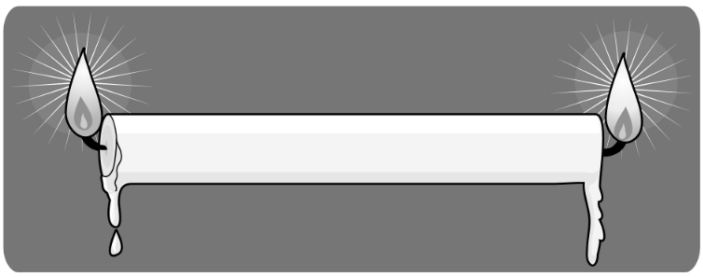
- 저장 공간 사용자들이 맞춤 스마트 계약 기능까지 사용해야 하는 대량의 복잡한 블록체인 거래들을 수행해야만 한다면 그것은 실행이 불가능할 정도로 너무 복잡한 일일 것이다. 이러한

복잡한 일을 중개 노드들에게 넘겨주면 저장 공간 사용자들은 통상 사용되는 이더리움 지갑을 통해 알파 노드로 지정된 중개 노드에게 Pearl을 한번만 전송하면 된다.

- 중개 노드들이 Pearl을 데이터 지도에 심도록 하면 교란 공격 벡터를 대폭 경감시킨다.

교란 공격이란 악의적인 저장 공간 사용자가 데이터 지도에 Pearl을 심는 척 하면서 실제로는 심지 않는 것을 말한다. 이 같은 사용자들이 Pearl이 하나도 없는 쓰레기 데이터를 업로드할 경우 이는 보물(실제로 존재하지 않는 보물)을 찾는 웹 노드들이 시간 낭비를 하도록 할 것이다. 결국 웹 노드들은 데이터 지도가 Oyster Protocol 사양의 범위 내에 들어 있지 않다는 것을 깨닫게 될 것이다. 하지만 그 시점에는 해당 웹 노드들이 사용한 에너지가 악의적 공격자가 투입한 에너지보다 클 것이다. 따라서 이는 해당 공격이 성공을 거두고 수익을 얻을 수 있도록 할 것이다. 그러나 웹 노드들이 중개 노드들을 통해 Genesis Hash(이들이 데이터 지도 전체를 정의한다)를 받게 되면 교란 공격은 대폭 경감된다. 왜냐하면 중개 노드가 무효한 데이터 지도(정확한 장소에 정확한 금액의 Pearl이 들어 있지 않은 데이터 지도)를 대표하는 Genesis Hash를 내놓기 시작하면 웹 노드들이 해당 중개 노드를 신고하는 것이 용이하고 이에 따라 해당 중개 노드의 평판은 손상되고 미래 트래픽이 줄어들게 되기 때문이다. 중개 노드들의 경우 그 신원 정보가 일정하고 여기에는 평판 정보까지 담겨 있다. 이에 비해 저장 공간 사용자들과 웹 노드들은 훨씬 더 역동적이다. 웹 노드들과 저장 공간 사용자들의 경우 일관된 암호화 신원 정보를 구성하는 것이 원래 어려울 뿐만 아니라 Oyster Protocol에서는 웹 노드들이 특정 규모의 보물 찾기를 수행할 때마다 매번 자시들의 신원을 리셋하도록 규정하고 있기 때문이다. 저장 공간 사용자들은 중개 노드들과 통신을 위해 세션에 연결되어 있는 경우를 제외하고 달리 분간할 수 있는 신원 정보가 없다.

저장 공간 사용자가 결제 용으로 Pearl을 제출하면 그 중 대략 절반 정도가 데이터 지도에 묻히고 나머지 절반을 두 중개 노드들이 보상으로 가져가게 된다. 데이터 지도에 Pearl을 묻는 이 두 노드들은 양쪽 끝에 불이 붙어 있는 초에 비유할 수 있다.



촛농은 데이터 지도를 나타내고 두 불꽃은 각각 중개 노드를 상징한다. 중개 노드 중 하나가 데이터 지도에 Pearl을 다 심고 난 후(초가 다 타고 난 후) 남은 Pearl을 가질 권리를 갖는다. 기본적인 경제적 관점에서 본다면 중개 노드가 데이터 지도에 Pearl을 매우 천천히 묻거나(초의 한쪽 만을 태우거나) 아예 묻지 않는 것이 유리하다. 만약 알파 노드가 초를 매초 10 단위씩 태우고 베타 노드가 초를 매초 2 단위씩 태운다면 이들은 여전히 결국에는 어느 지점에선가 만나게 되겠지만 베타 노드에게 남겨진 Pearl이 훨씬 많을 것이고 이것에 대해 베타 노드가 가질 권리가 있다. 이런 상황을 논리적으로 확장한다면 두 노드들이 초를 아예 태우지 않는 것, 즉 가장 많은 Pearl을 갖기 위해 노력하는 것이 경제적으로 가장 유리하다는 결론에 도달하게 된다.

분산 평판 시스템은 이 같은 경제적 인센티브를 정반대로 뒤집어 버린다. 중개 노드들에게 암호화된 신원이 배정되고 이들의 평판 점수는 가장 낮은 점수인 0에서 시작한다. 웹 노드들과 저장 공간 사용자들은 평판 점수가 가장 높은 중개 노드들과 거래를 하고자 한다(그러면서 동시에 지연과 다른 제약 사항들도 감안한다). 노드의 초가 타는 평균 속도가 산술적으로 증가하면서 노드의 평판 점수는 기하급수적으로 증가한다. 이로 인해 중개 노드들은 초를 더 빨리 태우기 위해 경쟁하게 된다. 비록 단기 간에는 Pearl 수입이 더 적음에 불구하고 말이다. 초를 가능한 발리 태우려 하는 중개 노드는 단기 간에 더 적게 벌지만 장기적으로는 기하급수적으로 더 많이 벌게 된다. 이에 따라 중개 노드들에게 있어 경쟁에서 이탈 하고자 하는 경제적인 요인은 없어지게 되는 것이다.

중개 노드가 Oyster Pearl을 데이터 지도에 묻으면 Oyster Contract의 특별한 파묻기 기능이 작동된다. 데이터 지도 상의 섹터 하나는 선택한 해쉬로부터 시작해서 백만 개의 해쉬(Genesis - N999,999, N1,000,000 - N1,999,999 등)을 나타낸다. 따라서 섹터 하나에는 임의의 GB 크기의 사용자 데이터가 담겨있다. 각 섹터 안에는 최소한 한 덩어리의 Pearl들이 묻혀 있어야 하지만 간혹 두 노드들 간에 조정이 불완전해서 두 덩어리의 Pearl들이 담겨 있는 경우도 생긴다. 섹터들 안에 Pearl의 위치는 두 노드들에 의해 임의로 결정된다. 따라서 섹터 하나 당 묻혀 있는 Pearl의 양은 해당 파일이 Tangle 상에 유지되는 기간을 결정한다. 1 PRL이 묻혀있다면 이는 임의의 GB 크기의 데이터가 Tangle 상에 1년 간 유지되도록 한다. 따라서 Oyster Contract는 원하는 저장 기간 동안 Pearl을 잠궈두는 기능을 한다. 이 기간 동안 웹 노드들은 이렇게 묻혀 있는 Pearl을 찾기 위해 작업 증명을 수행한다.

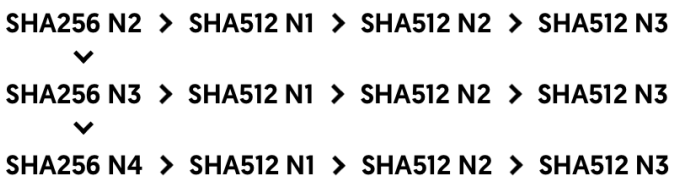
웹 노드가 Pearl은 요구하기 전에 해당 Pearl은 반드시 묻혀 있는 상태에 있어야 한다. 또한 다른 시간 대에서도 Pearl을 요구할 수 있는데 이런 시간 대를 Epoch라 칭한다. Oyster Protocol은 Epoch 하나를 1년 간 지속되는 것으로 정의한다. 이것은 데이터 지도 상 섹터 한 곳에 2 PRL 이 들어 있다면(저장 기간은 4년) 가용한 Epoch가 네 개, 즉 1년, 2년, 3년, 4년이란 뜻이다. Oyster Contract은 Epoch 하나 당 정확히 0.5 PRL 을 요구할 수 있도록 한다. 웹 노드들은 이들을 작동시킨 웹 사이트 소유자를 대신하여 Pearl을 요구한다.

Oyster Pearl 찾기

웹 노드들은 묻혀 있는 Oyster Pearl들을 찾기 위해 데이터 지도 상의 섹터들을 뒤지고 다닌다. 데이터 지도는 이른 바 Genesis Hash 라는 단일한 SHA256 해쉬 값으로 정의된다. 웹 노드들은 중개 노드들이나 다른 웹 노드들로부터 Genesis Hash를 얻는다. Genesis Hash는 공짜로 얻을 수 없고 이를 위해 하나의 웹 노드는 반드시 자신의 상대 노드가 정의한 작업 증명 임무들을 수행해야만 한다.
작업 증명 임무는 Tangle 상의 두 개의 미확정 거래들을 인용함으로써 정의되는데 이 중 하나가 가지로 나머지가 몸통으로 지정된다. 작업 증명이 완료되면 웹 노드는 자신이 제출한 거래의 신원 정보를 가지고 자신의 상대 노드에 응답을 하게 된다. 그러면 상대 노드는 Tangle을 확인해서 인용된 거래가 이전에 명시된 가지와 몸통 신원들과 일치하는지 검증하고, 그 거래가 그 거래 상에서 수행된 작업 증명을 포함하고 있는 데이터를 참조하는지를 검증한다. 상대 노드가 작업 증명이 수행됐음을 검증하면 이에 대한 대가로 Genesis Hash 가 전송된다.

Pearl 보물을 찾기 위해 웹 노드들은 새로 얻은 Genesis Hash로부터 유추된 데이터 지도 상의 임의의 섹터를 선택한다. 그 다음 웹 노드는 Tangle을 참조하여 현재 Epoch 에서 또 다른 웹 노드가 작업 증명을 수행했었는지를 확인한다. 그럴 경우 해당 웹 노드는 해당 섹터를 버리고 다시 다른 섹터를 시도한다. 왜냐하면 이 경우 해당 웹 노드가 해당 섹터의 Pearl을 찾았다 할지라도 다른 웹 노드가 이미 Oyster Contract 상에서 해당 Epoch에 대해 Pearl을 요구했을 가능성이 매우 높기 때문이다. 현재 Epoch에서 작업 증명이 이뤄지지 않았다면 웹 노드는 해당 섹터 내 각각의 연속적인 해쉬값들을 훑어보게 된다. 예를 들어 섹터 5를 선택했다면 업로드된 파일 중 임의의 GB 크기의 데이터를 대표하는 N5,000,000에서 N5,999,999까지의 해시값을 훑어보게 된다.
시간이 지나면 블러핑 (bluffing) 웹 노드 등과 같이 상기 기술된 작동 방식을 따르지 않는 새로운 웹 노드 전략들이 적용될 수 있을 것이다. 블러핑 웹 노드 방식이란 웹 노드가 특정 섹터에서 작업 검증을 하면서 해당 섹터에는 당분간 Pearl이 없을 거라고 허풍(블러핑, bluffing)을 치는 것이다. 이렇게 하면 다른 웹노드들이 해당 섹터에 접근하는 것을 막을 수 있게 되고 이로써 해당 섹터는 블러핑을 한 웹 노드가 독점할 수 있게 되는 것이다. 결국 블러핑을 구사한 웹 노드는 해당 섹터를 다음 Epoch가 나올 때까지 독점할 수 있을 것으로 기대하게 된다. 효과적으로 보물을 찾기 위한 보다 세련된 전략들이 만들어 지고 더욱 정교해짐에 따라 웹 노드들 간 상호 작용에서는 게임 이론의 원리가 더욱 복잡하고 고도화될 수 있는데 이는 결국 죄수의 딜레마로 이어지게 된다.

각각의 해쉬 값에 대해 웹 노드는 먼저 이 해쉬 값에 대응하는 거래에 대해 Tangle 상에서 작업 증명을 수행하고 이를 위해 장비의 GPU를 이용한다(WebGL2 사용). 따라서 해당 웹 노드는 해당 거래의 최대 1 KB 크기의 패이로드를 추출하여 현재 SHA256 해쉬의 SHA512 해쉬를 계산한다.



그 다음 해당 웹 노드는 SHA512 해쉬를 복호화 키로 사용하여 해당 페이로드의 잠금을 해제한다. 잠금이 풀린다면 해당 페이로드가 Pearl을 담고 있는 보물이라는 뜻이 된다. 풀리지 않으면 현재 SHA512의 SHA512 해쉬가 계산되고, 결국 해쉬체인이라는 일련의 연속된 정보 상의 다음 링크로 이동하게 된다. 만약 해당 웹 노드가 Oyster Protocol에서 정의해둔 해쉬체인 크기의 최대 한계치에 도달하면 해당 웹 노드는 데이터 지도 상의 다음 SHA256 해쉬로 이동하고 동일한 과정을 반복한다.

Oyster Protocol은 섹터 하나 당 보물이 최소 하나씩은 있어야 한다고 정의하고 있다. 만약 섹터 하나 전체에 보물이 하나도 발견되지 않으면 해당 데이터 지도는 무효로 선언되고 이 사항에게 분산 평탄 시스템 참가자들에게 경고 메시지가 전송된다. 이렇게 되면 해당 무효 데이터 지도에 대해 최초로 Genesis Hash를 소개했던 중개 노드의 평판이 저하되게 된다. 웹 노드 하나가 SHA512 해쉬체인 하나를 훑어 보는 데 수 일이 걸릴 수 있고, 따라서 섹터 전체에 대해서는 수 개월이 걸릴 수 있다. 이는 웹 노드들이 대량의 데이터를 갑자기 소비할 수 없도록 하는데 그 이유는 데이터 소비가 이처럼 소요되는 수 개월에 걸쳐 분산(하루 당 5MB 이하)되기 때문이다. 따라서 웹 노드들은 부담이 많이 되는 제한적이며 값비싼 데이터 접속 발생을 방지하기 위해 대역폭을 많이 사용하는 작업을 수행하지 않는다. SHA256와 SHA512 그리고 복호화 기능을 수행할 때는 모두 CPU 지시를 이용한다. 이는 웹 노드가 GPU를 통해 다른 웹 노드들 및 중개 노드들과 작업 증명을 위한 상호 작용을 수행할 수 있고 이와 동시에 CPU를 통해 데이터 지도 상에서 섹터 내 보물을 찾을 수 있다는 뜻이다.

Pearl을 데이터 지도에 묻을 때는 해당 Pearl을 담고 있는 이더리움 주소의 보안 시드 키(private seed key) 도 해당 Pearl과 함께 포함시킨다. 따라서 웹 노드가 보물을 찾으면 해당 웹 노드는 HTML5 localstorage 지시를 통해 해당 보안 시드 키를 저장하고 안전하게 지킨다. 이렇게 보물을 찾았음에도 불구하고 해당 웹 노드는 여전히 다음 두 가지 딜레마를 갖게 된다:

- 웹 노드들이 이더리움 블록체인에 직접 접속할 수 없고 그에 따라 복잡한 계약 기능들을 작동시키는 것이 어려울 것이다.
- 이더리움 주소에는 Pearl은 들어 있지만 가스 용 ETH는 들어 있지 않다. 따라서 먼저 어딘가로부터 해당 주소로 ETH를 전송해야만이 보안 시드 키가 거래를 생성하고 이를 이더리움 채굴자들이 접수하게 된다.

이 두 가지 딜레마로 인해 Oyster Protocol은 웹 노드가 중개 노드와 협업하여 보물의 잠금을 풀도록 정의하고 있다. 해당 웹 노드는 보안 시드 키를 통해 안전하게 보물을 전송하고 해당 중개 노드는 Pearl이 그 안에 진짜 포함되어 있는지를 확인한다. 일단 Pearl이 있다는 게 확인되면 해당 중개 노드는 매우 소량의 ETH를 이더리움 주소로 전송하여 거래에 필요한 가스로 기능하도록 한다. 그 다음 해당 중개 노드는 해당 거래를 블록체인에 전송한다. 이 거래가 Oyster Contract의 결제 요구 기능을 구동시켜 해당 Pearl을 해당 보물을 발견한 웹 노드와 통신하고 있는 웹 사이트 소유자의 이더리움 주소로 지급되도록 결제 요청한다. 그러나 여기에는 다음 두 가지 직접적인 우려 사항이 있다:

- 해당 웹 노드에게는 해당 중개 노드가 Pearl 지급을 요구할 때 해당 웹 노드와 통신하고 있는 웹 사이트 소유주의 이더리움 주소를 사용하지 않음으로써 해당 Pearl을 절취할 수 있다는 우려 사항이 있다.

- 해당 중개 노드는 보안 시드 키에서 추출된 이더리움 주소로 ETH를 전송해야 하는 데 이는 해당 웹 노드가 보물을 담고 있다고 주장하는 거래를 처리하는 데 가스로 사용하기 위해서이다. 그런데 이 웹 노드가 실제로는 악의적 공격자로서 웹 노드로 가장하여 해당 중개 노드를 속여서 소량의 ETH를 자신이 관리하는 이더리움 주소로 전송하도록 할 수 있다. 해당 중개 노드가 ETH를 전송하자 마자 해당 악의적 공격자는 최소 금액의 가스 결제가 이뤄진 후 차액을 절취한다. 이 차액은 매우 작을 수 있지만 이것이 무한히 반복되면 해당 중개 노드에 상당한 재정적 손실을 끼칠 수 있다.

이 같은 웹 노드 측의 우려 사항을 해결하기 위해 분산 평판 시스템은 해당 우려 사항을 발생시킨 중개 노드의 평판 등급을 신속히 저하시키도록 할 것이다. 중개 노드들은 일관된 암호화된 신원 정보를 유지하고 그 위에 평판 점수를 쌓아 나가며 이를 이용하여 웹 노드들과 저장 공간 사용자들이 해당 중개 노드를 이용하도록 설득한다. 웹 노드들은 보물을 발견하고 이를 잠금 해제할 때 가장 평판이 좋은 중개 노드들과만 함께 하고자 할 것이다. 만약 특정 중개 노드가 단일 데이터 지도 내에, 단일 섹터 내에서, 단일 Epoch 상의 Pearl을 절취하려 한다면 수익 보다는 훨씬 손해를 많이 보게 될 것이다. 해당 중개 노드는 몇 센트 얻으려다가 미래에 얻을 수 있는 수 천 달러를 날리는 꼴이 될 것이다. 따라서 해당 웹 노드는 평판이 좋은 중개 노드들과 믿고 거래할 수 있다.

중개 노드 측의 우려 사항을 해결하기 위해 중개 노드는 섹터 전체에 대해 작업 검증이 Tangle 상에서 최근에 완료됐다는 사실을 검증할 때 까지는 보물의 잠금을 해제해달라는 제3자로부터의 요청을 수락하지 않는다. 이는 악의적 공격자가 중개 노드를 속여서 보물의 잠금을 해제하도록 하려면 그 보다 먼저 섹터 전체에 대해 작업 검증을 수행해야만 한다는 것을 의미한다. 이러한 악의적 공격의 시도가 얼마나 헛된 노력인지 비유를 하나 들자면 다음과 같다. 악의적 공격자는 1센트 벌기 위해 작업 검증 퀴즈를 푸는 데 전기료로 5달러를 내야만 한다.
따라서 만약 중개 노드가 보물이 있는 섹터에 대해 작업 검증이 완료됐음을 확인했다면 이는 ETH를 Oyster Contract 의 결제 요청 기능을 구동시키기 위해 이더리움 주소로 전송하는 것이 경제적 가치가 있다는 것이다.

웹 노드와 중개 노드 간 협업

Oyster 생태계 내에서 발생하는 주요 상호 작용들 중 하나는 웹 노드들이 다른 웹 노드들과 중개 노드들로부터 정부를 구매하고 이에 대해 결제하기 위해 다량의 작업 증명을 수행하는 것이다. 따라서 웹 노드들은 올바르게 작동하기 위해서 Tangle과 연결을 지속할 수 있어야 한다. 언젠가는 웹 노드들이 Tangle에 직접적으로 접속하는 것이 가능해지겠지만 지금으로서는 현재의 라이브러리와 하드웨어/대역폭 제약으로 인해 웹 노드들은 Tangle 네트워크 상에서 라이트 클라이언트로서 기능하도록 제약을 받고 있다. 이는 웹 노드들이 Tangle 요청과 제출을 처리해줄 중개 역할을 하는 라이트 클라이언트 호스트를 필요로 한다는 것을 의미한다. Tangle 라이트 클라이언트 호스트가 이미 중개 노드들과 별도로 독립적으로 존재하지만 이들 대다수를 웹 노드들이 사용할 수 없는 데 그 이유는 이 호스트들이 SSL을 통해 요청을 처리할 수 없기 때문이다. Oyster Protocol은 중개 노드들이 모든 Tangle 요청을 SSL을 통해 처리하도록 요구한다. 이는 Oyster Protocol을 구동할 대다수의 웹 사이트들이 SSL을 통해 호스팅될 것으로 예상하기 때문이고, 그에 따라 웹 노드 구동 로직을 반드시 SSL을 통해 로딩시켜야 하고 모든 수신이나 송신 통신도 SSL을 통해 이뤄져야만 하기 때문이다.

중개 노드들은 또한 웹 노드들이 다른 웹 노드들과 peer-to- peer 연결을 통해 직접적으로 상호 작용하도록 한다. 이를 위해 PeerJS Library 가 사용되는 데 이는 WebRTC Standard를

기반으로 하고 있다. 따라서 중개 노드는 PeerJS 서버를 구동시켜서 웹 노드들이 직접 서로 간에 통신을 할 수 있도록 한다.

웹 노드들은 늘 Genesis Hash들, 그 중에서도 새로운 Genesis Hash들을 요구하게 되는 데 그 이유는 첫 번째 Epoch의 묻혀 있는 Oyster Pearl들에 대해 소유권 주장이 이뤄지지 않기 때문이다. 일단 중개 노드가 저장 공간 사용자와 파일 업로드 세션을 완료하면 해당 중개 노드가 Genesis Hash를 보유하게 된다. Oyster 네트워크가 통상적인 평형 상태에 도달하면; 중개 노드들은 항상 새로운 Genesis Hash들을 과도하게 공급받게 되는 한편 웹 노드들은 Genesis Hash들에 대해 과도한 수요를 갖는 상태가 된다.

중개 노드들은 새로운 Genesis Hash들을 무료로 나눠주지 않고 그 대신 Genesis Hash를 주는 대가로 대량의 작업 증명을 요구한다. 이렇게 하는 1차적 이유는 악의적 사용자들이 쉽게 Genesis Hash를 획득하고 여기에 묻혀 있는 보물을 찾는 것을 방지하기 위함이다. 따라서 이렇게 작업 증명 부담을 추가함으로써 악의적 사용자들이 중개 노드들로부터 Genesis Hash를 구하는 것을 더욱 더 경제성이 떨어지게 하는 효과를 갖는다. 이러한 작업 증명 요구는 또한 Oyster Protocol 이 지양하는 이타적 활동을 제거하는 역할도 한다.

중개 노드가 웹 노드에게 부과하는 작업 증명 부담은 저장 공간 사용자가 해당 중개 노드에게 부담시키는 것과 동일한 작업이다. 따라서 만약 Oyster 네트워크의 평형이 완벽하게 이뤄지면 중개 노드들은 작업 증명을 수행하지 않을 것인데 그 이유는 이 작업들이 지속적으로 웹 노드들에게 상쇄될 것이기 때문이다. 웹 노드들과 중개 노드들 간 발생하는 교환 시퀀스는 웹 노드들 간에 벌어지는 것과 동일하다. 교환 시퀀스는 다음과 같다:

- 웹 노드가 중개 노드에게 새로운 Genesis Hash나 믿을 수 있는 이웃 웹 노드 신원(Neighbor identity)들을 갖고 있는지 묻는다(한 번에 한 가지 종류의 정보만 요청한다).

- 해당 중개 노드가 응답한다. 그리고 이 경우 있다고 표시를 한다. 해당 중개 노드는 또한 요청된 작업 증명의 작업 규모를 나타내야 한다. 작업 규모는 Oyster 네트워크 상 수급 상태에 따라 변동한다.

- 해당 웹 노드가 제안한 작업 증명 규모에 동의하면 해당 작업을 수락한다고 응답한다.

- 해당 중개 노드는 세 개의 거래에 대한 세 개의 참조를 Tangle 상에 전송한다. 이들 거래 중 하나에는 해당 중개 노드가 이전에 처리하기로 되어 있었던 할당됐던 유관 저장 공간 사용자 데이터가 담긴다. 나머지 두 개의 거래는 미확정 거래들로서 IOTA Algorithm 으로 확정되도록 하는 것이 바람직하다. 이 둘은 각각 가지와 몸통 거래로 명시된다.

- 해당 웹 노드가 Tangle 상에서 replay Bundle 기능을 수행하고, 이에 따라 해당 중개 노드가 지정한 그 대로 수동으로 거래의 가지와 몸통을 설정한다.

- 일단 해당 웹 노드가 작업 증명을 완료하고 Tangle 거래 전체가 제출되고 나면 해당 웹 노드는 이렇게 새로 제출된 거래의 신원을 해당 중개 노드로 전송한다.

- 해당 중개 노드는 해당 거래 신원이 실제 Tangle 네트워크 상에서 올바른 데이터를 대표하는지, 그리고 앞서 명시된 가지와 몸통 할당이 올바른지를 검증한다.

- 합의된 작업 증명 규모를 충족시키기 위해 작업 증명을 추가로 수행해야 할 경우 위에 기술한 절차대로 수행된다.

- 일단 합의된 작업 증명 규모가 충족되면 해당 중재 노드는 이에 대한 대가로 Genesis Hash나 이웃 웹 노드 신원을 해당 웹 노드에게 전송한다. 해당 중개 노드는 분산 평판 시스템에 의해 주기로 합의한 정보를 제공하지 않을 수 없는 압박을 받는다.

시간이 흐름에 따라 Genesis Hash들이 중개 노드들로부터 웹 노드들의 집단 의식으로 이주한다. 웹 노드들은 모든 섹터의 모든 Epoch에 대해 소유권 주장이 이뤄지면 오로지 의도적으로만이 Genesis Hash를 망각할 수 있다. 이 것은 데이터가 원래 만료되도록 되어 있고 따라서 해당 저장 공간 사용자가 더 많은 Pearl을 추가함으로써 해당 데이터의 수명을 연장시키지 않는 한 해당 데이터가 작업 증명에 의해서 더 이상 보장되지 않는 시점이 온다는 것을 나타낸다. 웹 노드들은 데이터와 이들에 의해 알려진 Genesis Hash를 유지하는 데 HTML5 localstorage 지시를 사용한다. 해당 localstorage 지시에 의해 제공된 공간이 꽉 차면 해당 웹 노드는 해당 웹 사이트 소유자 입장에서 수익성 전망이 가장 낮은 Genesis Hash를 삭제함으로써 데이터를 정리하기 시작한다.

Genesis Hash들이 중개 노드들에서 웹 노드들로 이주하는 것은 필수적인 요소이다. 저장 공간 사용자와 업로드 세션이 완료되면 Genesis Hash는 해당 세션의 두 개 중개 노드들에만 일시적으로 존재한다. 웹 노드들이 Genesis Hash를 찾게 됨에 따라 발생하는 긍정적인 결과는 그로 인해 Genesis Hash가 Oyster 네트워크 내에 확실하게 존재할 수 있도록 한다는 것이며, 그에 따라 중개 노드들만이 Genesis Hash를 독점 보유하게 되는 최초 위험을 제거해준다는 것이다. Genesis Hash가 Oyster 의 집단 의식에 의해 잊혀지게 되면 해당 Genesis Hash는 더 이상 작업 증명을 통해 유지되지 않는다. 따라서 Tangle은 해당 데이터를 노드들 내에 장기 간에 걸쳐 유지할 책임을 더 이상 지지 않게 된다.

웹 노드들 간 상호 작용

웹 노드들은 서로 peer-to-peer 방식으로 상호 작용을 하는 데 그 이유는 Oyster 네트워크 경제 체계 내에 존재하는 수급 제약 때문이다. 이 같은 peer-to-peer 연결은 WebRTC Standard를 기반으로 하는 PeerJS Library 를 통해 이뤄진다. 웹 노드들이 서로 통신을 하기 위해서는 Oyster 네트워크 상에서 상호 간에 식별이 가능해야만 한다. 따라서 각 웹 노드는 암호화된 의사 영구 신원을 채택한다. 의사 영구 신원들이란 웹 노드가 보물을 찾는 일을 하는 과정에서 마치 Oyster 시스템에 새롭게 들어오게 된 것과도 같이 자신의 메모리를 다 지우고 처음부터 다시 시작해야 하는 단계에 도달할 때까지만 신뢰할 수 있고 일관성을 갖도록 되어 있다는 뜻이다. 이렇게 하는 하는 것은 Oyster 네트워크 상의 웹 노드들 간에 역동적인 회전 주기를 유도해내기 위해서이다. 웹 노드들이 동일한 이웃 웹 노드들과 무한대로 계속해서 통신을 하고자 한다면 Oyster 네트워크는 과도하게 정적이게 되고 환경 변화에 반응하지 않게 될 것이다. 웹 노드들 중 대다수가 Oyster Protocol의 신원 갱신 규약을 따르는 한 소수의 웹 노드들 중에 상당 기간 동안 동일한 이웃 웹노드들과 연결을 유지하기를 기대하는 웹 노드들은 그런 기대를 접도록 강제하게 된다.

웹 노드가 Oyster 네트워크에 새로 들어 왔거나 최근에 자신의 신원을 갱신했을 경우 해당 웹 노드는 이웃 웹노드가 없게 되고 그렇기 때문에 이웃 목록을 구축해야 한다. 이로서 해당 웹 노드는 Catch 22 딜레마에 빠지게 된다. 이웃 신원들은 다른 웹 노드들이 공유한다. 하지만 해당 웹 노드는 처음에 이웃 웹 노드가 없기 때문에 이웃 웹노드에게 요청할 수 없다. 이에 대한 최초 해결책은 중개 노드에게 요청하는 것이 맞을 것으로 보이지만 중개 노드 신원들 조차도 웹 노드들이 참여하는 분산 평판 시스템을 통해 공유된다. 따라서 웹 노드들은 최초의 신뢰받는 중개 노드를 기본 참조값으로 받아들인다. 웹 사이트 소유자들은 이 기본 참조값을 자신들이 신뢰할만하다고 생각하는 중개 노드들 중 아무 거로나 바꿀 수 있는데 그에 따라 네트워크의 분산 원칙을 준수하게 된다.

중개 노드들은 이미 서로 간의 신원을 알고 있는 웹 노드들 간에 최초 연결을 중개하는 데 사용된다. 따라서 중개 노드는 최근에 활성화 되어 있는 웹 노드들과 이들의 신원들을 보유하고 있다. 신규 웹 노드가 이러한 신원들을 중개 노드들로부터 구매하고 그 대가로 교환 시퀀스에 정의되어 있는 작업 증명을 수행한다. 신규 웹 노드는 중개 노드들로부터 신원들을 지속적으로

구매하면서 동시에 자신이 새로운 관계를 맺은 다른 웹 노드들로부터도 신원들을 구매한다. 이에 따라서 해당 신규 웹 노드의 이웃 웹 노드 목록은 기하급수적으로 확장되고 이것은 수확체감의 법칙에 따라 포화 단계에 이를 때가지 계속된다. 웹 노드가 지나치게 많은 이웃 웹노드를 만드는 것은 바람직하지 못한 데 그 이유는 (작업 검증을 통해) 이웃 웹노드를 찾는 데 소모한 에너지를 Genesis Hash를 구매하고 보물을 찾는 데 대신 쓸 수 있었기 때문이다. 이웃 웹 노드 목록이 확장됐기 때문에 해당 웹 노드는 이제 분산 평판 시스템으로부터 평판 명세서를 충분히 받을 수 있다.
따라서 해당 웹 노드는 새로운 중개 노드들과 통신할 수 있고 또한 기존의 중개 노드들도 유지할 수 있다. 웹 노드 신원들은 주기적으로 리셋되기 때문에 웹 노드들은 지속적으로 그러나 점진적으로 새로운 이웃 웹 노드들을 찾지 않을 수 없다.

웹 노드는 다른 웹 노드들 및 중개 노드들과 교환 시퀀스를 수행할 수 있다. 이로 인해 해당 웹 노드는 작업 검증을 하고 그 대가로 Genesis Hash나 이웃 웹 노드 신원들 등과 같은 귀중한 정보를 얻게 된다. 다음은 교환 시퀀스에 대한 설명이다:

- 웹 노드가 이웃 웹 노드에게 새로운 Genesis Hash나 이웃 웹 노드 신원(Neighbor identity)들을 갖고 있는지 묻는다(한 번에 한 가지 종류의 정보만 요청한다).

- 해당 이웃 웹 노드가 응답한다. 그리고 이 경우 있다고 표시를 한다. 해당 이웃 웹노드는 또한 요청된 작업 증명의 작업 규모를 나타내야 한다. 작업 규모는 Oyster 네트워크 상 수급 상태에 따라 변동한다.

- 해당 웹 노드가 제안한 작업 증명 규모에 동의하면 해당 작업을 수락한다고 응답한다.

- 해당 이웃 웹 노드는 세 개의 거래에 대한 세 개의 참조를 Tangle 상에 전송한다. 이들 거래 중 하나에는 해당 이웃 웹 노드가 이전에 처리하기로 되어 있었던 할당됐던 유관 저장 공간 사용자 데이터가 담긴다. 나머지 두 개의 거래는 미확정 거래들로서 IOTA Algorithm 으로 확정되도록 하는 것이 바람직하다. 이 둘은 각각 가지와 몸통 거래로 명시된다.

- 해당 웹 노드가 Tangle 상에서 replay Bundle 기능을 수행하고, 이에 따라 해당 이웃 웹 노드가 지정한 그 대로 수동으로 거래의 가지와 몸통을 설정한다.

- 일단 해당 웹 노드가 작업 증명을 완료하고 Tangle 거래 전체가 제출되고 나면 해당 웹 노드는 이렇게 새로 제출된 거래의 신원을 해당 이웃 웹 노드로 전송한다.

- 해당 이웃 웹 노드는 해당 거래 신원이 실제 Tangle 네트워크 상에서 올바른 데이터를 대표하는지, 그리고 앞서 명시된 가지와 몸통 할당이 올바른지를 검증한다.

- 합의된 작업 증명 규모를 충족시키기 위해 작업 증명을 추가로 수행해야 할 경우 위에 기술한 절차대로 수행된다.

- 일단 합의된 작업 증명 규모가 충족되면 해당 이웃 웹 노드는 이에 대한 대가로 Genesis Hash나 이웃 웹 노드 신원을 해당 웹 노드에게 전송한다.

통상 중개 노드들로부터 발생한 Genesis Hash에 대한 작업증명 부담의 규모가 다른 웹 노드들로부터 발생한 Genesis Hash에 대한 작업증명 부담보다 더 크다. 그 이유는 중개 노드들에는 상대적으로 신규 Genesis Hash들이 더 많은 반면 웹 노드들은 상대적으로 더 오래된 Genesis Hash를 갖고 있기 때문이다. 신규 Genesis Hash의 시세는 오래된 것에 비해 더 높을 것으로 예상되는데 그 이유는 신규 Genesis Hash에 아직 찾지 못한 보물이 있을 것이라는 기대가 더 크기 때문이다. 이것은 또한 웹 노드들이 다른 웹 노드들에게 Genesis Hash를 요청하기 전에 중개 노드들에게 먼저 Genesis Hash를 요청한다는 것을 암시하는

것이다. 바로 이 점이 Genesis Hash들이 지속적이고 효과적이며 빠르게 중개 노드들로부터 데이터 손실로 인한 영향을 거의 받지 않는 웹 노드들의 집단 의식 쪽으로 이동할 수 밖에 없도록 만드는 메커니즘인 것이다.

웹 노드들은 상호 간에 통신을 할 때 지속적인 통신 유형들에 대해 연결 지연 정도를 측정한다. 즉 일정한 크기의 페이로드 교환을 수행하는 모든 통신/거래들에 대해 연결 시작에서부터 종료까지의 소요 시간을 측정한다는 뜻이다. 따라서 웹 노드는 자신과 인접해 있는 이웃 웹 노드들 간의 대략적 상대 거리를 줄일 수 있게 된다. 이러한 연결 지연 정보를 계속 유지하여 시간이 지남에 따라 점진적으로 웹 노드가 멀리 떨어진 이웃 웹 노드들 보다는 인접해 있는 이웃 웹 노드들을 선호하도록 한다. 이 같은 행동의 결과로 시간이 지남에 따라 특정 웹 노드의 경우 자신의 이웃 웹 노드들 목록이 최적화되어 자신과 인접해 있는 웹 노드들과 일차적으로 통신하게 된다. 이와 동일한 최적화가 특정 웹 노드의 중개 노드들 목록에도 이뤄진다. 비록 중개 노드의 경우 지연보다는 평판이 더 중요시 되긴 하지만 말이다.

이 같은 통신 지연 최적화의 결과로 Oyster 네트워크는 써드 파디 어플리케이션을 구축하는 기반이 되는 효과적인 노드 홉 경로(node hop pathway)를 갖춘 분산형 저지연 메시 네트워크가 된다. 예를 들어 숙련된 프로그래머들로 이뤄진 소집단들이라면 누구든 코어 웹 노드 프로토콜 로직을 확장시키고 자신의 이더리움 토큰을 사용하는 분산형 자바 스크립트 전화 서비스를 작성할 수 있다. 따라서 이런 확장 기능이 Oyster 커뮤니티 내에서 오픈 소스로 공개하고 공유될 수 있다. 따라서 웹 사이트 소유자들은 이 같은 전화 서비스 하위 경제로부터 창출되는 추가 수입을 얻기 위해 이 같은 확장 기능을 추가할 수 있다. 이는 창조적인 컨텐츠 공급자들이 웹을 통해 수익을 올릴 수 있는 역량을 더욱 향상시켜주고 그에 따라 Oyster 네트워크의 목적이 달성된다. 이러한 전화 통화 메커니즘은 웹 노드들이 제공하는 API 콜 상에서 간단하게 구동되기 때문에 음성 패킷들을 Oyster 네트워크 상에 원하는 수신자에게 효과적으로 전송할 수 있게 된다. 따라서 Oyster 프로토콜은 간단한 API를 통해 최적화된 메시 네트워크와 자동화된 노드 홉 로직 통신을 제공함으로써 분산 코드 개발과 적용의 기초 토대 역할을 하는 확장 플랫폼으로 설계된다.

컨텐츠 소비 권한

인터넷 상의 기본적인 사회적 계약은 정보의 교환이다. 웹 사이트 보유자들은 원천 컨텐츠 그리고/또는 서비스들을 생성/획득/제공하기 위해 투자를 한다. 이들은 호스팅 비용도 부담해야 한다. 이들이 이 같이 투자를 하려면 경제적 반대 급부가 있어야만 한다. 공짜 점심은 없듯이 말이다.

이러한 반대 급부 필요를 충족하기 위해 인터넷은 그 동안 광고 교환이라는 그저 그런 방식에 의존해 왔다. 광고는 어디서나 지속적으로 사용자의 주의를 산란하게 하고, 컨텐츠와 관련성이 없으며, 사생활을 침해하고, 웹 사이트의 디자인 연속성을 파괴한다. 따라서 광고는 인터넷 전체에서 모두의 경멸 대상이 되었다. 이에 대한 당연한 대응으로 광고 차단 프로그램들이 주류를 이루게 되면서 인터넷 경제에 손해를 초래했다. 이에 따라 창조적인 컨텐츠 공급자들은 여전히 컨텐츠 제작과 호스팅 비용을 충당해야만 하는 상황에서 이러지도 저리지도 못하는 처지에 이르게 됐다. 시간이 흐름에 따라 창조적인 컨텐츠 공급자들은 중앙집중화된 광고 교환소의 정책, 의사결정, 변덕에 휘둘리게 되었다.

Oyster Protocol은 낡은 광고 패러다임을 탈피하는 새롭고 파격적인 방식으로서 창조적인 컨텐츠 공급자들이 자신의 컨텐츠로 수식을 올리는 것에 대해 완전한 자율권을 가질 수 있도록 한다. 웹 사이트 방문자들은 입장료만 내면 되고 짜증나고 아무 관계 없는 내용들로 인해 방해 받을 필요가 전혀 없게 된다. 여태 고생해온 창조적 컨텐츠 공급자들에게 돈이 흘러가기 때문에 컨텐츠의 양과 품질의 저하가 멈추고 다시 한번 향상될 수 있다. 이는 다시 방문자들이 사이트를 계속 방문하도록 그에 따라 Oyster Protocol 을 통해 자신들의 컴퓨터 자원을

사용하도록 유도하게 된다.

웹 사이트 소유자가 Oyster Protocol을 사용하는 방법은 매우 간단하다. 자신들의 웹 사이트 HTML에 아래의 코드 한 줄만 추가하면 Oyster Protocol 을 모두 사용할 수 있고 Pearl로 결제를 받을 수 있게 된다.

<script id="o.ws" data-payout="ETH_ADDRESS"
src="https://oyster.ws/webnode.js"></script>

웹 사이트 방문자들이 컨텐츠 이용 대가로 자신들의 컴퓨터 자원을 제공하는 데 동의하지 않을 경우 이들은 간단하게 Oyster Protocol을 비활성화시킬 수 있다. 이 경우 비활성화된 해당 웹 노드의 HTML5 localstorage 영역에 차단 플래그가 설치된다. 그러면 해당 방문자의 장비는 어떠한 작업 증명이나 보물 찾기도 수행하지 않게 된다. 그러나 해당 웹 사이트 소유자의 사이트에 자바스크립트 플래그가 활성화되어 해당 사용자가 참여하지 않음을 표시한다. 따라서 웹 사이트 소유자는 보물 찾기 작업에 동의하지 않는 방문자라면 누구에게든 쉽게 컨텐츠 전송을 차단하도록 선택할 수 있다.

웹 노드들은 데이터를 저장하는 데 HTML5 localstorage 지시를 사용하기 때문에 다른 웹 사이트 소유자들에 의해 구동되는 동안에도 동일한 신원과 작업 순서(work queue)를 유지한다. 일단 보물 찾기 세션이 개시되면 해당 웹 노드는 자신을 구동시킨 웹 사이트 소유자의 이더리움 주소를 청구인으로 영구적으로 지정한다.

예를 들어 어떤 사람이 노트북으로 즐겨찾는 웹 사이트들 중 네 곳을 둘러 보는 데 이 중 두 곳에 Oyster Protocol 이 활성화되어 있다고 치자. Oyster Protocol이 활성화되어 있는 A 웹사이트를 방문하면 해당 방문자의 노트북이 웹 노드가 되고 모든 활성화된 보물 찾기를 A 사이트의 소유자에게 귀속시킨다. 따라서 발견된 모든 Pearl에 대해 Oyster Contract 에 따라 A 사이트의 이더리움 주소로 결제를 청구하게 된다. 그 다음 같은 사람이 Oyster Protocol이 활성화되어 있는 B 웹사이트를 방문한다. 이 사람의 노트북은 여전히 웹 노드 역할을 하고 동일한 암호화된 신원을 유지하며, Genesis Hash와 다른 웹 노드들 및 중개 노드들의 신원들, 그리고 현재 보물을 찾고 있는 모든 데이터 지도 정보를 보유한다. B사이트의 관할 하에 새로운 보물 찾기가 시작되면 이에 따라 발견된 Pearl은 B 사이트의 소유자에게 귀속된다.

Oyster Pearl 토큰 기능

Oyster 네트워크는 완전히 분산된 시스템이기 때문에 해당 네트워크에서 사용되는 가치 토큰인 Oyster Pearl을 관리하기 위한 무신뢰 메커니즘을 필요로 한다. Oyster Pearl은 이더리움 블록체인 상에서 운용되는 ERC20 호환 토큰으로서 Oyster Protocol 기능을 활성화하기 위한 사용자 설정 속성을 담고 있다.

Pearl 토큰에 적용된 사용자 설정 기능은 파묻기 기능(bury function) 이다. 이더리움 주소를 파묻음으로써 Pearl 입금이 이뤄지는 동안에는 Pearl 출금이 이뤄지는 것을 차단한다. 묻혀져 있는 주소로도 여전히 입금은 가능한데 이는 저장 공간 사용자들이 자신들의 데이터에 대한 수명을 연장하여 의도적인 데이터 기간 만료를 방지하게 한다. 최초로 파일을 Tangle에 업로드할 때 중개 노드들이 Oyster Contract의 파묻기 기능을 구동시킨다. 해당 중개 노드가 데이터 지도에 묻어 놓은 Pearl들이 해당 Oyster Contract에 의해 인출되고 그에 따라 지출할 수 없게 된다.

웹 노드들이 보물 찾기를 하는 과정에서 파묻기 기능이 구동되어 있는 이더리움 주소들의 보안 시드 키들과 마주치게 되는데, 그에 따라 Pearl들을 잠궈서 이들이 일시에 인출되지 못하도록 한다. 따라서 보안 시드 키를 회수한 주체들은 그것이 웹 노드이든 아니든 관계 없이 그 누구도 모든 ERC20 호환 토큰들 상에서 구동되는 일반적인 이체 기능을 통해 Pearl을 조금도 인출할 수

없게 된다. 웹 노드는 중개 노드에게 웹 사이트 소유자의 이더리움 주소를 대신하여 결제 청구를 구동시켜달라고 요청해야만 한다. 해당 결제 청구 기능은 묻혀 있는 상태에 있는 이더리움 주소에 의해서만 구동될 수 있다. 해당 Oyster Contract가 특정 섹터에 대한 Epoch들을 계산하고 하나의 Epoch에 해당하는 양만큼의 Pearl을 청구자에게 할당한다. 하나의 섹터 내에 결제 청구가 되지 않은 Pearl이 두 개 이상의 Epoch 분량만큼 있다면 청구자에게 이들 모두를 보상으로 지급한다. 해당 Oyster Contract은 섹터 메트릭스는 감안하지 않아도 되는데 그 이유는 묻혀 있는 각각의 이더리움이 이미 정확히 섹터 하나만을 대표하기 때문이다.

중개 노드가 보물을 찾은 웹 노드를 구동시켰던 청구권을 가진 웹 사이트 소유자의 이더리움 주소로 결제 청구 기능을 구동시킨다. 해당 청구 기능은 또한 수수료 주소 변수도 정의한다. 해당 중개 노드가 해당 결제 청구 기능을 구동시키면 해당 중개 노드는 자신의 이더리움 주소를 수수료 변수로 제출한다. 따라서 Oyster Contract는 해당 중개 노드에게 보물 잠금을 풀어주고 획득한 수수료를 자동으로 할당한다. 따라서 중개 노드들이 받는 수수료는 만장 일치로 결정되며 감사도 가능하다. 해당 결제 청구 기능이 수행되면 청구 대상 Pearl 전액이 해당 웹 사이트 소유자의 이더리움 주소로 전송되고 이 중 합의된 비율만큼이 중개 수수료로 중개 노드에게 배정된다.
따라서 Oyster Pearl은 웹 사이트 소유자, 웹 노드, 중개 노드, 저장 공간 사용자의 경제적 동기를 서로 연결해주는 필수적인 교환 매체이다.

파일 검증 및 추출

Oyster Protocol을 통해 파일을 업로드하려 하면 저장 공간 사용자의 클라이언트는 해당 데이터를 Tangle에 전송할 중개 노드 두 개를 선택한다. 해당 파일의 시작 부분과 끝 부분 두 곳에서부터 데이터를 처리하는 해당 두 개의 중개 노드들이 각각 한 부분 씩을 맡아서 처리하는데 이는 마치 촛불을 양 끝에서 태우는 것과 같다. 두 중개 노드들은 어느 시점엔가 데이터 지도의 중간 즈음 어디에선가 만나게 된다. 그러나 발생할 가능성은 별로 없는 경우이긴 하지만 두 중개 노드 중 하나가 작업 증명을 하지 않고 이탈한다면(그리고 Oyster Pearl은 자기가 가진다면) 이탈하지 않는 중개 노드가 끝까지 데이터 처리를 완료하게 된다. 해당 저장 공간 사용자의 클라이언트는 이 같은 이탈 사실을 파악했을 것이고 암호화를 통해 이탈한 중개 노드를 분산 평판 시스템을 통해 보고했을 것이다.

일단 해당 파일이 모두 Tangle에 전송되면 해당 클라이언트는 데이터 지도 전체를 다운로드 하여 해당 지도의 무결성을 검증한다. 해당 클라이언트는 Tangle에 접속해서 해당 데이터 지도를 다운로드할 때 최초 (파일 업로드 때 썼던) 두 개 중개 노드가 아닌 다른 중개 노드들을 사용한다. 이 검증 단계는 기술적으로 생략할 수 있으나 수행할 것을 권장하는데 그 이유는 가능성은 매우 희박한 경우이긴 하지만 두 개의 중개 노드들이 해당 저장 공간 사용자에 대항해서 공모하거나 각자 이탈을 할 수 있고, 이 경우 해당 저장 공간 사용자의 클라이언트가 이 같은 위반 사실을 분산 평판 시스템을 통해 암호화 형태로 신고하기 때문이다. 이 경우 정직한 중개 노드들이 부정직한 중개 노드들이 하지 않은 일을 하기로 결정하고 그러면서 Oyster 사용자로부터는 어떠한 Pearl 결제도 요청하지 않는다. 정직한 중개 노드들이 이 같은 데이터 교정 작업을 수행하는 이유는 이를 통해 자신들의 평판이 대폭 올라가고 이로 인해 장래 수입 향상 전망이 밝아지기 때문이다.

검증 절차는 부정직한 중개 노드들을 막아내는 기능과 함께 프로그램 에러나 실행 버그로 인한 데이터 지도 상 오류가 절대 발생하지 않도록 보장하는 기능을 한다. 다음은 업로드 검증과 데이터 회수에 사용되는 다운로드 시퀀스에 대한 설명이다.

- 클라이언트가 Oyster Handle로부터 Primordial Hash를 불러온 다음 이를 SHA256 함수에 제출하여 Genesis Hash를 생성한다.

- 클라이언트가 SHA256 해쉬 체인에서 선택된 해쉬의 삼진법 형태 데이터를 계산한다 (첫 번째 반복 처리를 위한 Genesis Hash).

- 클라이언트가 앞 단계에서 생성된 삼진법 데이터와 상관 관계에 있는 Tangle 거래의 페이로드 데이터를 추출한다. 이 추출 작업은 (웹 노드들과 저장 공간 사용자들이 사용하는)분산 평판 시스템에 따라 선택된 중개 노드를 통해 이뤄진다. 검증 절차를 수행하는 경우 이를 위해 Tangle에 접속하는 중개 노드들은 최초 파일 업로드에 사용된 중개 노드들과 동일하면 안 된다.

- 해당 페이로드 데이터가 추출되면 클라이언트는 Oyster Handle 전체를 암호화 키로 사용하여 해당 데이터의 잠금을 해제를 시도한다. Oyster Protocol은 또한 암호화 시스템에 암호문구도 쓸 수 있도록 하는 데 저장 공간 사용자가 해당 암호문구를 잊어버릴 수 있기 때문에 주의를 요한다. Oyster Handle이나 선택 사항인 암호문구를 분실하면 해당 파일을 영구적으로 분실하게 된다.

- 해당 페이로드 데이터의 잠금이 해제되면 그것은 업로드된 파일을 구성하는 데이터 시퀀스의 일부분이다. 만일 잠금 해제가 안되면 그것은 묻혀 있는 보물을 담고 있는 SHA512 해쉬체인에 대한 참조이다. 클라이언트는 데이터 지도를 훑어나가면서 섹터 당(1,000,000 SHA256 해쉬) 최소 한 개의 SHA512 해쉬체인 참조가 있는지 확인하고 그렇지 않을 경우 해당 데이터 지도를 무효로 선언하고 분산 평판 시스템과 적합한 절차를 수행한다.

- 일단 해쉬 한 개의 페이로드 데이터를 추출되면 이는 저장 공간 사용자의 영구 저장소에 저장되고 메모리 사용 할당으로부터 자유로워진다.

- 클라이언트는 현재 SHA256 해쉬를 SHA256 함수로 제출함으로써 SHA256 해쉬체인에 다음 반복 처리를 계산한다. 그 결과 나온 해쉬는 해쉬체인의 차기 반복처리가 된다. (예, N1 해쉬는 Genesis Hash 다음에 온다).
- 해당 파일 전체가 Tangle로부터 추출될 때까지 상기 절차가 반복된다.
- 분할되었던 파일의 여러 부분들이 다시 결합되고 전체 컨텐츠를 내장되어 있는 검사합계와 비교하여 데이터 무결성을 확보한다.

저장 공간 사용자는 아무 클라이언트나 사용하여 파일에 접속할 수 있고 심지어 중개 노드가 아닌 Tangle 노드로부터도 가능하다. 해당 데이터를 추출하는 데 필요한 단 두 가지는 Oyster Handle과 IOTA Tangle에 대한 일반적인 접속 이다.

분산 평판 시스템

저장 공간 사용자들은 Pearl을 묻기 위해 중개 노드를 필요로 하고 Pearl을 청구하기 위해 웹 노드를 필요로 한다. 이 두 경우 모두 Pearl은 중개 노드에 전송되어 처리된다. 따라서 저장 공간 사용자들과 웹 노드들은 중개 노드들을 감시하기 위해 모니터링 시스템을 이용한다. 이 시스템이 바로 분산 평판 시스템인데 그 작동 방식은 eBay와 비슷하다. 중개 노드들은 eBay의 판매자들과 같고 웹 노드들/저장 공간 사용자들은 eBay의 구매자들과 같다. 웹 노드들/저장 공간 사용자들은 자신들이 찾을 수 있는 한 평판이 가장 좋은 중개 노드들과만 거래를 한다. 중개 노드 선택 알고리듬은 네트워크 지연, 트래픽 제약, 프로토콜 금지사항(예, 저장 공간 사용자들은 파일 업로드와 검증 작업에 동일한 중개 노드를 사용할 수 없다) 등과 같은 다른 기준들도 고려한다. 이와 같은 별도의 기준들이 있음에도 상대적으로 소규모의 최고 수준의 평판을 가진 중개 노드들이 트래픽의 절대 다수를 받고 그에 따라 Pearl 수익을 올리게 된다. 따라서 중개 노드들은 수익을 얻기 위해서는 정직을 지키는 것이 필수적이다.

모든 평판 점수는 0에서 시작한다. 마이너스 평판 점수는 없는데 그렇지 않으며 악의적인

중개 노드가 자신의 마이너스 평판 점수를 없애기 위해 0점에서 시작하는 새로운 암호화된 신원을 생성할 수 있기 때문이다. 이렇게 평판 점수를 바꿔치기 하는 잘못된 사기 수법은 계속 새 계정을 만드는 가짜 eBay 판매자에 비교할 수 있다. 평판 점수가 전혀 없는 신규 계정들은 사람들의 관심을 거의 받을 수 없게 된다. 대신 평판이 높은 정직한 판매자들이 절대 다수의 수익을 가져가게 된다.

0점에서 시작해서 정직한 중개 노드들은 웹 노드들과 저장 공간 사용자들이 Tangle에 접속하도록 중개해줌으로써 천천히 점진적으로 발전한다. Tangle에 대한 접속이 진짜인지 검증 가능하며 내고장성을 갖고 있는 데 이는 좀 더 평판이 높은 노드들을 참조하여 Tangle에 진짜 접속했는지 여부를 확인하기 때문이다. 정직한 중개 노드가 최초 명성을 충분히 획득하면 웹 노드들/저장 공간 사용자들로부터 가치 기반의 거래(Pearl을 묻거나 찾는 것)를 수행해 달라는 요청을 받기 시작한다. 이들은 최초 베타 노드로 지정되는데 이는 베타 노드들이 취급하는 Pearl 가치가 더 낮기 때문이다. 이 같은 가치 기반 거래들이 올바르게 수행되면 해당 중개 노드의 명성이 기하급수적으로 올라간다. 이에 따라 더 많은 웹 노드들/저장 공간 사용자들이 해당 중개 노드에게 가치 기반 거래 처리를 요청하게 된다. 부정직한 중개 노드들은 기하급수적으로 명성이 올라가는 지점에 도달하는 경우가 거의 없다.
각 중개 노드들의 신원은 생성된 PGP 키를 기반으로 하는데 이것은 절대 공개되지 않도록 보관해야 한다. PGP가 유출되면 이는 해당 노드의 평판(미래 수익 전망)을 급속히 떨어뜨리는데 그 이유는 악의적 사용자들이 단기 수익을 노리고 해당 키를 앞다투어 사용하기 때문이다. 중개 노드는 가장 먼저 PGP 키를 안전하게 지킬 수 있는 역량을 갖춰야 한다. 이게 선행되고 나서야 중개 노드는 데이터 지도 상의 보물 위치 정보와 보물의 보안 시드 키를 안전하게 지킬 수 있는 능력이 있음이 확인되는 것이다.

중개 노드들과 웹 노드들/저장 공간 사용자들은 가치 기반 거래를 수행하기로 합의하기 전에 다음 두 가지 조건을 협의한다; Tangle의 최소 상태와 블록체인의 최소 상태. 원장의 최소 상태는 계약이 성공적으로 이행되고 계약 마감일이 오기 전에 반드시 있어야 하는 거래의 최소 범위를 정의한다. 저장 공간 사용자의 클라이언트는 데이터 지도 상 각각의 SHA256A에 대한 가지와 몸통 거래들을 Tangle이 보유해야만 하는 것의 최소 범위로 정의한다. 이에 대해 해당 중개 노드는 자신이 관리하는 주소로 이체되는 Pearl을 포함하기 위한 블록체인의 최소 범위를 설정한다. 만약 제안된 계약이 Oyster Protocol과 계약 양 당사자들의 환경적 맥락에 맞으면 두 당사자는 자신들의 PGP 서명으로 계약서에 전자 싸인을 한다.

중개 노드들은 항상 데이터 지도들을 쌍으로 설치하여 이 둘이 가장 많은 Pearl을 심도록 경쟁을 시킨다. 이와 대조적으로 각 노드는 저장 공간 사용자의 클라이언트에 의해 가지/몸통으로 지정된다. 일단 데이터 지도 설치가 완료되면 관찰을 하고 있는 웹 노드들은 계약 상에 정의된 몸통/가지 지정 정보를 Tangle 상에 참조되어 있는 실제 몸통 가지들과 비교하여 각 중개 노드의 실적을 예측한다. 이에 따라 분산 평판 시스템 참가자들 사이에 성과가 제일 좋은 주체가 누구인지에 대해 만장일치의 합의가 이뤄지게 된다. 가장 많은 작업 검증을 수행한 중개 노드가 평판 점수 업그레이드를 받는다. 한편 소수의 작업 증명만을 수행한 중개 노드는 (성과의 정도에 따라) 평판 점수에 변화가 없거나 점수가 떨어지게 된다. 따라서 작업 증명을 최대한 신속히 수행하는 것이 장기적으로 중개 노드에게 (수익 창출 측면에서) 이롭다. 이러한 경제적 압박에 의해 저장 공간 사용자는 신속하고 효과적으로 파일을 업로드할 수 있게 된다.

중개 노드들은 Genesis Hash나 웹 노드 신원을 구매하고자 하는 웹 노드들에게 작업 증명 부담을 넘길 수 있다. 따라서 평판이 좋은 중개 노드는 통상 데이터 지도 설치를 더 빠르게 하는데 그 이유는 더 많은 웹 노드들로 더 많은 교환 시퀀스를 가질 수 있고 그에 따라 가지/몸통 작업 증명 지정을 웹 노드들에게 더 많이 넘길 수 있기 때문이다. 만약 계약 마감일 이후에 서명된 계약서에서 비롯된 몸통/가지 지정 중 Tangle과 일치하는 것이 전혀 없거나 거의 없는 경우 이는 이탈로 간주되고 해당 중개 노드의 평판은 심각한 손상을 입게 된다.

웹 노드가 발견된 보물의 잠금을 풀어야 할 경우 해당 웹 노드는 계약서 상에 블록체인의 최소 상태를 정의한다. 해당 최소 상태는 보물의 지도에서 나온 Pearl에 대한 결제 청구는 반드시 Oyster Contract 상에 웹 사이트 소유자의 이더리움 주소로 이뤄져야 한다고 명시한다. 이 계약에 대해서 Tangle의 최소 상태는 정의되지 않는다. 해당 중개 노드가 계약 조건에 동의하면 해당 중개 노드는 자신의 PGP 키를 써서 계약서에 전자 서명한다. 해당 웹 노드가 서명된 계약서의 사본을 받아야만이 해당 웹 노드는 보물의 보안 시드 키를 중개 노드에게 전송한다. 블록체인의 최소 상태가 계약 마감일 전까지 충족되지 않으면 해당 웹 노드는 서명된 계약서를 참조하여 해당 사실을 다른 분산 평판 시스템 참가자들에게 통보한다. 따라서 엡 노드들로 구성된 메쉬 네트워크는 해당 중개 노드가 해야 할 일을 하지 않았다는 것에 점차적으로 의견의 일치를 보게 되고 이에 따라 해당 중개 노드의 평판과 장래 수익 창출 전망을 떨어뜨리게 된다.

결론

Oyster Protocol은 수익 창출과 익명의 접속 가능한 저장 공간 그리고 분산 어플리케이션 개발 및 적용을 해결하기 위해 고안됐다. 이더리움 블록체인이 토큰 생성을 위한 간단명료한 기본 틀을 제공하는 것과 마찬가지로 Oyster Protocol은 분산 메쉬 네트워크에 접속할 수 있는 간단명료한 기본 틀을 제공한다.

일상 생활에서 사용하는 컴퓨터, 스마트폰, 자동차, 냉장고를 포함하여 웹 브라우저가 설치되어 있는 것이라면 무엇이나 웹 노드가 될 수 있다. 이들은 상호 간에 직접 통신을 하고 이따금씩 중개 노드들로부터 연결 중개를 필요로 할 뿐이다. 시간이 경과하면서 이들은 지연이 낮은 이웃하는 웹 노드들을 자동으로 선택함으로써 네트워크 전체적으로 연결이 최적화 상태에 도달하게 된다. 널리 알려져 있고 사용하기 쉬운 언어인 자바 스크립트로 확장 프로그램을 만들 수 있음으로 개발자들은 전세계에 펴져 있는 메쉬 네트워크에 접속할 수 있다. 이는 분산 어플리케이션들을 쉽게 구축하고 최적화된 고성능에 지연이 최적화된 메쉬 네트워크에 접속할 수 있도록 하는 최고의 환경을 조성하도록 한다.

Oyster Protocol은 수백 만개의 웹 사이트들이 수익 창출 잠재력을 실현시킬 수 있도록 하고, 개인과 기업의 데이터 저장 문제를 해결할 수 있도록 하며, 개발자들이 필요로 하는 메쉬 네트워크 플랫폼을 제공해줄 수 있게 한다.