

# Ara — 지급, 유저 아이덴티티, 호스팅, 스트리밍, 그리고 디지털 파일과 콘텐츠 소유권을 위한 글로벌 분산형 구조 (초안)

Eric Jiang, Charles Kelly, Joseph Werle, Tony Mugavero, Vanessa Kincaid

마지막 업데이트 2018년 11월 14일 (일부\*\*)

## 개요

인터넷은 더 이상 예전과 같은 모습이 아니며 전 세계 정보의 흐름과 비용, 소비 방법 및 시기를 제어함으로써 모든 정보에 전권을 행사하는 몇몇 대기업들이 지배하게 되었습니다. Ara는 분산형 플랫폼이자 이를 해결하기 위해 설계된 프로토콜 세트입니다. 새로운 소유권 증명 시스템을 사용하여 디지털 파일 및 콘텐츠를 라이선스하고 판매함으로써 Ara는 글로벌 데이터 전달을 처리하고 고유의 Ara 토큰으로 이러한 자산을 구매할 수 있도록 지원합니다. 또한, Ara 플랫폼은 이 모든 것의 실행을 통해 유비쿼터스 및 분산 사용자 ID와 지갑 시스템을 활용하여 사용자가 개인 정보 소유권을 유지할 수 있게 합니다. 사실상, Ara는 인터넷에 존재하는 정보를 호스팅하고 제공하는 방법과 소비자가 정보를 사용하고 비용을 지불하는 방법을 다루는 새롭고 현대적인 정신적 모델입니다. Ara는 네트워크에 참여하고 호스팅함으로써 보상을 받는 시스템에 기여할 수 있다는 점에서 기업뿐만 아니라 소비자를 위한 새로운 패러다임을 불러일으켰습니다. P2P(Peer-to-Peer) 파일 공유, 소유권 및 라이선스를 위한 블록체인 기술, 분산 컴퓨팅은 모두 분산형의 효율적인 단일 시스템으로 결합됩니다.

수없이 많은 구성원이 Ara의 혜택을 누리고 있습니다. 소비자는 컴퓨팅을 위한 에어비앤비와 비슷한 Ara 토큰을 얻기 위해 유류 스토리지, 대역폭, 처리 능력을 사용할 수 있으며 이러한 토큰을 사용하여 콘텐츠를 구입할 수 있습니다. 기업은 P2P 기술을 통해 이러한 정보를 사람들에게 제공하는 것을 줄임으로써 소비자와 다른 기업에 대한 비용을 절감합니다. 누구든지 네트워크의 데이터 센터로 참여하여 보상을 얻을 수 있습니다. 디지털 크리에이터, 게임 및 소프트웨어 개발자, 영화 및 TV 스튜디오, 게시자는 Ara 토큰을 사용하여 네트워크에 라이선스를 받은 콘텐츠를 게시하여 더 많은 수익 공유를 창출하고 호스팅 비용을 위해 나머지를 팬에게 제공함으로써 그들의 작품에서 더 많은 수익을 올릴 수 있습니다. 바로 모두에게 이익이 되는 원-원 상황입니다. 소비자는 보상을 받고, 게시자는 더 많은 돈을 벌고 기업은 수익성이 향상됩니다. 이 모든 것은 분산화되고 중립적인 방식으로 이루어지므로 어떠한 중간 회사도 여러분에게 정보와 콘텐츠를 제공하는 단일 사업을 방해할 수 없습니다.

\*\* 이것은 2018년 6월부터 백서에 업데이트된 부분입니다. 앞으로 몇 달 안에 더 완전한 업데이트가 배포될 예정입니다.

참고: Ara는 활발한 연구 개발을 진행 중입니다. 이 문서의 내용은 변경될 수 있습니다. 최신 버전은 <https://ara.one>에서 볼 수 있습니다. 의견이나 제안 사항이 있으면 [hello@ara.one](mailto:hello@ara.one)으로 문의하십시오.

# 목차

<b>1. 소개</b>	<b>2</b>
1.1 배경 . . . . .	2
1.2 개요 . . . . .	2
1.3 플랫폼 서비스 . . . . .	3
<b>2. 플랫폼 개요</b>	<b>4</b>
2.1 AraID . . . . .	4
2.1.1 분산 ID . . . . .	4
2.2 분산 콘텐츠 전달 네트워크(DCDN) . . . . .	5
2.2.1 Ara 파일 시스템(Ara File System (AFS)) . . . . .	5
2.2.2 토큰 사용법 . . . . .	5
2.3 Ara 프로토콜군 . . . . .	7
2.3.1 보상 및 인센티브 . . . . .	7
2.3.2 파일 전달 . . . . .	8
2.3.3 스마트 계약 . . . . .	8
<b>3. 향후 개발</b>	<b>10</b>
3.1 모듈 . . . . .	10
3.2 Ara 명칭 시스템(Ara Name System (ANS)) . . . . .	10
<b>4. 감사의 말</b>	<b>10</b>
<b>약자</b>	<b>11</b>
<b>참고 문헌</b>	<b>11</b>
<b>부록</b>	<b>13</b>
<b>I. 완전히 성장한 Ara 플랫폼 토큰 경제(초안)</b>	<b>13</b>
개요 . . . . .	13
Ara 토큰 . . . . .	13
기능 . . . . .	14
시장 역학 . . . . .	14
인센티브 구조 . . . . .	14
네트워크 효과 . . . . .	15
참고 문헌 . . . . .	16
<b>II. DCDN 비용 분석 by Lester Kim</b>	<b>17</b>
업로더의 수익 최대화 . . . . .	17
배포자의 비용 최소화 . . . . .	19
예제 . . . . .	21
참고 문헌 . . . . .	22
<b>III. 예상되는 Ara 보상 분석 by Lester Kim</b>	<b>24</b>
네트워크 모델 . . . . .	24
예제 . . . . .	25
참고 문헌 . . . . .	25

# 1. 소개

## 1.1 배경

오늘날의 하이퍼미디어 환경은 구식입니다. 애그리게이터와 앱 스토어는 콘텐츠 생성자에 대한 지배력을 강화하고 있습니다. 기존의 콘텐츠 전달 네트워크는 비효율적이고 비용이 많이 들며, 클라우드 컴퓨팅은 소수의 선택된 게이트키퍼에게 집중되어 있습니다. 그리고 데이터는 데이터를 소유한 사람이 아니라 이를 통해 수익을 창출하는 사람이 저장합니다. 콘텐츠 개시자와 생성자는 가격을 억지로 부풀려야 하기에 이렇게 느리고 비싼 시스템에 대한 비용을 소비자에게 떠넘기므로 개시자, 소비자, 생성자 모두에게 가치를 잃고 있습니다.

2021년까지 동영상이 전체 인터넷 트래픽의 80% 이상이 될 태세이며 콘텐츠 비용은 파일 크기와 콘텐츠 제공 비용이 증가하면서 계속 상승해 왔습니다 [3]. 4K, VR, AAA 게임 모두 이러한 추세에 기여하고 있습니다. 소비자는 거래형 콘텐츠와 구독을 위해 비용을 더 지불하고 있을 뿐만 아니라 무료 콘텐츠를 보기 위해 복잡하고 폭력적인 광고 시스템을 마주해야 합니다. 이러한 요소들은 저작권 침해 문제를 악화시켜 콘텐츠 소유자들에게 수십억 달러의 손실을 초래하므로 광고를 건너뛰거나 제거하기 위해 광고 차단기와 같은 도구를 도입하는 결과를 가져옵니다 [16][12][4]. 이로 인해 소비자는 광고를 기피하고 콘텐츠 소유자는 손실된 수익을 만회하고자 광고 차단기를 차단하는 차단기를 만들고 더 비싼 구독 서비스로 이어지게 됩니다. 이는 악순환입니다.

P2P(Peer-to-peer) 파일 배포 아키텍처는 이러한 비효율성에 대한 대응으로 나타났으며 Napster와 같은 중앙 집중식 서버를 통합한 하이브리드 솔루션에서 Gnutella, 최종적으로 BitTorrent와 같은 완전한 분산형 솔루션으로 진화하고 있습니다. 오늘날 P2P 파일 전달은 비용 측면에서 매우 효율적이므로 Microsoft와 같은 기업은 Windows 10 배포 비용을 절감하기 위해 자체 Azure 인프라가 아닌 이 방식을 사용합니다.

그러나 P2P 파일 공유 네트워크는 비용 측면에서 효율적이기는 하지만 공공 환경에서 사용되었을 때 무임승차, 저작권 침해, 해킹 및 암시장으로 넘쳐나는 역사를 가지고 있습니다. 개별적으로 업로드되고 있는 콘텐츠에는 그러한 행위를 할 권리가 있다는 믿음이 없었으며 씨드된 콘텐츠가 콘텐츠 소유자의 의도대로 전달되었는지 확인할 수 있는 방법도 없었습니다. 또한, 콘텐츠를 저장하고 공유하는 것에 대한 보상도 없었으므로 사용자는 전달 시스템에 씨드를 유지해도 받을 수 있는 인센티브가 없었습니다. 피어들은 자신을 위해 콘텐츠를 받고 다른 피어들에게 콘텐츠를 씨딩하는 데 계속해서 참여하지 않았습니다. 이를 해결하기 위해, P2P 아키텍처들은 바티 전략, 평판 시스템, 독점 화폐와 같은 인센티브 메커니즘을 통합하기 시작했습니다. 그러나 이러한 메커니즘에도 나름의 공유 문제가 있으며 Sybil 및 Whitewashing 공격의 대상이 되기도 합니다.

## 1.2 개요

이 백서에서는 커뮤니티 주도의 탈중앙형 및 분산형 컴퓨팅이면서 콘텐츠 제공 플랫폼인 Ara를 소개합니다. Ara를 사용하면 미사용 프로세싱, 스토리지, 대역폭 용량을 활용하여 동시에 전 세계의 모든 장치가 글로벌 슈퍼컴퓨터, 데이터베이스, 전달 네트워크의 일부가 될 수 있습니다. 이러한 장치와 함께 누구나 참여해서 이익을 얻을 수 있는 생태계인 Ara 네트워크를 형성합니다.

네트워크는 기본적으로 서로 겹치는 소비자, 서비스 요청자, 서비스 제공자, 소프트웨어 개발자로 구성되며 각각은 채택을 위한 자체 인센티브를 가지고 있습니다. Ara

를 이용하면 서비스 요청자는 그 어느 때보다도 끊임없이 확장되고 있는 분산 서비스 라이브러리와 함께 방대한 양의 컴퓨팅 리소스를 손쉽게 확보할 수 있습니다. 스마트 폰, 노트북, 게임 콘솔 기기에 이미 비용을 지불한 서비스 공급업체는 사용하지 않는 리소스를 임대하여 수익을 창출하기 시작할 수 있습니다. 보상을 받기 위한 유일한 요건은 해당 계정에서 헬드 역할을 하는 소액의 보증금을 제출하는 것입니다. 이러한 헬드는 언제든지 철회할 수 있지만 아무 보상이라도 받고 교환해야 합니다. 소프트웨어 개발자는 Ara 생태계의 유례없는 규모를 활용하여 컴퓨팅 작업을 대량으로 수행하고 요청자와 제공자가 참여하는 새로운 분산형 서비스를 만들 수 있습니다. 한편, 소비자는 자신이 좋아하는 공연을 보고 음악을 듣는 것에 대해 모두 보상을 받으면서 그들의 일상 생활을 영위할 수 있습니다.

따라서 예비 컴퓨팅 리소스를 보유한 누구라도 즉시 서비스 이행자로서의 역할을 수행하여 콘텐츠 배포 지원에 대한 보상을 받을 수 있고 원격 리소스를 찾는 사람은 Ara의 분산 서비스를 요청할 수 있으며 기존 클라우드 컴퓨팅 서비스 제공업체와 비교하여 적은 비용으로 향상된 보안, 파일 가용성, 전달 속도를 제공할 수 있습니다. Ara는 인프라 구입 및 관리의 부담을 덜어주므로 음반 회사를 거치지 않고 자신의 새 앨범을 자유롭게 직접 게시할 수 있는 인디 아티스트에서부터 고객에게 다가가기 위해 더 이상 애그리게이터를 거칠 필요가 없는 대규모 미디어 대기업에 이르기까지 모든 유형의 콘텐츠 생성자가 혜택을 누릴 수 있습니다. Ara는 네트워크 구성원이 제공하는 리소스에 의존합니다. 네트워크가 커질수록 더 견고하고 효율적으로 작동합니다.

### 1.3 플랫폼 서비스

Ara 플랫폼은 3가지 핵심 서비스와 시스템으로 구성됩니다.

1. **AraID:** AraID는 Ara 플랫폼에 있는 모든 에이전트와 콘텐츠에 대해 안전하고 분산되고 검증 가능한 글로벌 아이덴티티를 수립하여, 권리를 가진 소유자에게 데이터 제어 권한을 돌려줍니다.
2. **Decentralized Content Delivery Network (DCDN):** DCDN은 콘텐츠 무결성, 인센티브, 버전 관리, 분산 ID를 지원하는 기본 피어투피어, 보안 분산 파일 시스템, 스토리지 네트워크(AFS)에 대한 Ara의 네트워크 역할을 합니다.
3. **Protocol Suite:** Ara는 DCDN, AraID, 이더리움 블록체인 사이의 신뢰할 수 없는 상호운용성을 가능하게 만들어 주는 안전한 프로토콜 제품군을 통해 연결됩니다.

## 2. 플랫폼 개요

무충돌 파일 시스템 네트워크 (Conflict-Free File System Network (CFSNet))은 Ara의 피어투피어 분산 파일 시스템, AraID, 분산형 콘텐츠 전달 네트워크 (Decentralized Content Delivery Network (DCDN))의 근간입니다. CFSNet은 기본적인 Merkle 트리 구조와 정확한 이벤트 프로토콜의 동기화 가능한 장부(Syncable Ledger of Exact Events Protocol (SLEEP)) [1] 파일 형식을 활용하여 기존의 파일 전달 방식, 즉 클라이언트 서버 및 P2P 모두에게 있는 많은 우려 사항을 해결하고 베전 관리와 개정 이력뿐만 아니라 암호화가 보장된 콘텐츠 무결성을 제공함으로써 IPFS와 같은 기존 기술을 개선합니다. 네트워크는 CFS라고 불리는 격리된 파일 시스템으로 구성되어 있습니다. 그뿐만 아니라 각각의 CFS 인스턴스는 파티션을 지원하고, 각각의 디렉토리가 자체 액세스 레벨이 있는 자체 완비된 CFS 아카이브로 존재할 수 있도록 파일시스템 계층 표준 (Filesystem Hierarchy Standard (FHS))의 하위 집합을 구현합니다. 이러한 파티션 중에 AFS는 /home 및 /etc 파티션을 사용하여 AFS 콘텐츠, 메타데이터를 각각 저장합니다. 각각의 CFS 파티션은 생성 시에 생성된 고유한 Ed25519 32바이트 공개 키를 사용하여 네트워크 전체에서 공개적으로 식별할 수 있습니다. CFS의 공개 키는 해당 파일 시스템에 읽기 전용 액세스 권한을 부여하며 개인 키 소유자만이 그 안에 포함된 콘텐츠를 업데이트하고 게시할 수 있습니다.

### 2.1 AraID

AraID는 Ara 플랫폼의 모든 사용자와 콘텐츠에 대해 안전하고 검증 가능한 분산형 표현을 생성하고 해결해야 할 책임이 있습니다. W3C 분산형 식별자 (Decentralized Identifier (DID)) [15] 사양을 완벽하게 준수하는 AraID는 사용자 및 콘텐츠를 표현하기 위해 DID 설명자 객체 (DID Descriptor Object (DDO))를 사용합니다(그림 1 참조). DDO들은 소유자가 제어하는 서비스 엔드포인트와 개인 커뮤니케이션 채널을 포함하여 인증 및 인기뿐만 아니라 기타 ID 속성 방법을 정의하는 간단한 JSON-LD 문서입니다 [15]. DDO는 절대로 개인 식별 정보 (Personally-Identifiable Information (PII))를 저장하지 않으므로 이러한 서비스 엔드포인트와 커뮤니케이션 채널은 이를 얻는 안전한 방법을 파악하여 엔티티가 개인 정보

및 온라인 ID에 대한 자기 주권을 가질 수 있도록 합니다 [15].

#### 2.1.1 분산 ID

AraID는 Ara 플랫폼의 모든 사용자와 콘텐츠에 대해 다음과 같은 형태로 생성됩니다:

- did:ara:ee93189c629cdaf94  
9fd57bac5b005b916936d2a5c6806  
40fd1aedc8315730a0

AraID는 DIF(Decentralized Identity Foundation) 시스템의 일부로 DID(위의 ara)의 두 번째 구성 요소로 표시되는 Universal Resolver 메서드를 구현합니다 [11]. 이 메서드는 드라이버라고도 알려져 있으며 Ara 플랫폼에서 DID와 DDO가 해결되는 방법을 정의합니다. 인터넷 URI와는 다르게 DID는 등록이나 제어를 위해 중앙 권한을 요구하지 않으며 TCP/IP 및 DNS에서 찾을 수 있는 비-단사, 비-전사 관계보다는 DDO와 일대일 대응을 형성합니다.

AraID 보안의 핵심은 분산 공개 키(DPKI)를 [13] 사용하여 암호화된 방식으로 유지되며, 여기서 Ed25519 공개 키는 DID(ee9318...위)의 id부분과 해당 DDO가 저장되는 CFS의 공개 키, 양쪽 모두로 사용됩니다. 이러한 문서에는 디지털 서명, 암호화, 기타 암호화 작업에 사용되는 다양한 키를 담고 있는 publicKey 속성이 포함되어 있습니다. ID가 생성되면 이러한 배열은 소유하고 있는 ID의 키를 비롯하여 해당하는 이더리움 계정의 공개 키로 채워집니다.

AFS AraID의 경우에는 관련된 콘텐츠의 메타데이터가 포함된 /etc 파티션의 공개 키도 저장됩니다. 이에 대한 키는 AFS DDO에 저장되므로 AFS DID를 보유하고 있는 어떠한 요청자라도 이를 해결할 수 있습니다.

새로운 ID가 생성될 때마다 키쌍을 씨드하기 위해 니모닉 구문이 사용됩니다. 니모닉은 개인 키를 안전하게 보관하고 쉽게 유지하기 위해 소유자에게 반환되고 개인 키가 필요 없이 엔티티가 DID의 소유권을 쉽게 검증하고 계정을 복원할 수 있게 합니다.

#### ID 보관 및 해결

ID가 생성되면 처음에는 로컬에 쓰여지므로 네트워크에 복귀하기 전에 모든 로컬 해결이 캐시를

확인할 수 있습니다. 그러나 원격으로 ID를 해결하려면 먼저 ID를 보관해야 합니다. Ara는 향후의 해결을 위해 이러한 ID를 저장하는 작업인 아카이버 노드를 실행합니다.

아카이버 노드와 마찬가지로 Ara는 요청된 DDO에 대한 아카이버를 쿼리하는 역할을 하는 리졸버 노드도 실행합니다. 리졸버는 문제의 AraID를 보관했던 원격 아카이버의 네트워크에 접근하기 전에 디스크에 저장될 수 있는 ID를 로컬에서 해결하기 위해 먼저 확인할 것을 요청합니다.

```
{
  'ddo': {
    '@context': 'https://w3id.org/did/v1',
    'id': 'did:ara:ee9318...',
    'authentication': [
      {
        'type': 'Ed25519SignatureAuthentication2018',
        'publicKey': 'did:ara:ee9318...#owner'
      },
      {
        'publicKey': [
          {
            'id': 'did:ara:ee9318...#eth',
            'type': 'Secp256k1VerificationKey2018',
            'owner': 'did:ara:ee9318...',
            'publicKeyBase58': 'H3C2AVvLMv6gmMNam...'
          }
        ],
        'service': {
          'ens': 'https://etherscan.io/enslookup',
        }
      ...
    }
  }
}
```

도표 1: DDO 예시

### 이더리움 계정

각각의 ID는 이더리움 계정 및 연결된 이더리움 지갑으로 생성되며 ID를 생성하는 동안 생성된 임의의 니모닉을 사용하여 복구할 수 있습니다. 이러한 니모닉을 사용하여 이더리움 계정과 ID 자체가 결정론적으로 생성되므로 사용자는 단지 이 니모닉을 이용하여 자신의 이더리움 계정과 지갑을 포함한 완전한 ID를 복구할 수 있습니다.

AraID는 공개 키 암호화로 보호받는 모든 계정을 지원하도록 설계되었습니다 따라서, 지원할 수 있는 암호화폐 계정의 유형에 대해 불가지론적이며 어떠한 종류의 암호화폐와도 쉽게 연결될 수 있습니다.

## 2.2 분산 콘텐츠 전달 네트워크 (DCDN)

DCDN은 확장 가능한 분산형 하이퍼미디어 및 디지털 자산 배포를 위한 Ara의 솔루션입니다.

DCDN의 핵심은 콘텐츠와 관련 메타데이터를 보관하는 CFS 구현, Ara 파일 시스템(AFS)의 네트워크로 구성되어 있다는 점입니다.

### 2.2.1 Ara 파일 시스템(Ara File System (AFS))

AFS는 Ara의 특정 요구와 목표를 충족할 수 있도록 맞춤화된 CFS의 특징입니다. AFS는 CFS가 구현한 기존의 두 개 파티션인 /home 및 /etc 파티션을 활용합니다. FHS의 [8] 하위 집합으로 구현된 이러한 파티션은 원시 이진 데이터와 콘텐츠 메타데이터를 각각 담당합니다. /home 파티션은 사용자가 AFS 콘텐츠를 구입했거나 액세스 권한을 부여받은 후에만 액세스할 수 있지만, 메타데이터가 포함된 /etc 파티션은 콘텐츠 소유권과 관계없이 액세스할 수 있습니다. AFS 소유자는 요청자가 파싱할 수 있도록 메타데이터를 위한 스키마를 정의 할 수도 있습니다. 이 프로토콜은 메타데이터를 구조화하는 방법에 관한 엄격한 표준을 실시하지는 않습니다. 하지만 분산형 서비스 사이의 상호운용성을 최상으로 지원하기 위해 기존의 패러다임에 대한 참고 사항으로 Schema.org을 추천합니다.

AFS가 처음 생성될 때 AraID는 BIP39의 [5] 12개 단어로 구성된 무작위의 니모닉 구문을 사용하여 생성됩니다. 생성된 DID는 AFS의 공개 키로 사용되며, 해당 DDO의 인증 속성은 소유자의 DID를 포함할 수 있도록 수정됩니다. 그렇게 함으로써 DID를 해결하여 AFS 소유자를 판별할 수 있습니다.

AFS는 시스템에 도입된 각 콘텐츠에 대해 생성됩니다. 이것은 영화와 같은 단일 파일이거나 게임과 같은 파일의 모음일 수 있습니다. 콘텐츠 소유권을 암호화된 방식으로 확인하기 위해 두 세트의 바이트 버퍼가 이더리움 블록체인에 기록됩니다. 첫 번째는 `metadata.tree` 항목이며 이는 데이터 스토리지 계층 내에 포함된 데이터의 직렬화된 머클 트리를 나타냅니다. 두 번째는 직렬화된 트리의 루트 노드의 서명이 포함되어 있는 `metadata.signatures` 파일입니다.

### 2.2.2 토큰 사용법

DCDN과 관련하여 Ara 토큰을 소유하면 처음에는 다음과 같은 기능이 부여됩니다.

1. 네트워크 내에서 콘텐츠를 구매하고 다운로드할 수 있는 기능.
2. 훌드 역할을 하는 Ara 보증금을 제출하여 P2P 파일 전달에 참여하고 모든 콘텐츠에 대한 보상을 받을 수 있는 기능.

## 2.3 Ara 프로토콜군

다음의 하위 단원에서는 플랫폼의 핵심 프로토콜을 정의하고, 시스템의 각 부분을 상세히 설명하고, 그들 간의 상호운용성을 설명합니다.

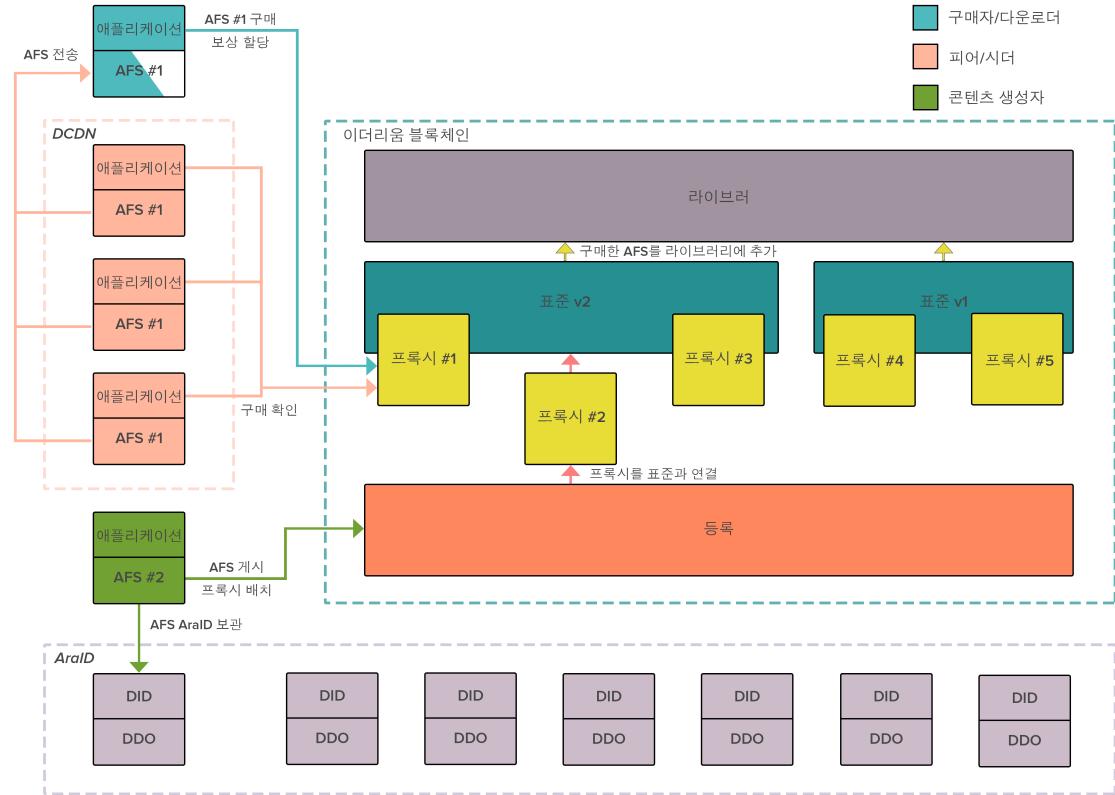


도표 2: Ara 프로토콜의 그림

### 2.3.1 보상 및 인센티브

BitTorrent와 같은 기존의 피어투피어 파일 공유 시스템은 이타적 행위에 의존하며 피어가 다운로드하는 만큼 네트워크에 업로드하기 위한 효과적인 인센티브가 부족하기 때문에, *leechers*(배포된 파일을 다운로드하는 사용자)가 *swarm*(배포된 파일을 다운로드하거나 업로드하는 모든 피어)를 쉽게 지배할 수 있는 불균형을 초래합니다 [7]. 이러한 불균형은 무임승차 문제[6]의 한 형태이며 leecher는 종종 그 대가로 아무것도 제공하지 않고 다른 사람들의 희생으로 네트워크에서 이익을 얻기 때문에 건강한 네트워크에서는 바람직하지 않습니다. Ara는 이러한 네트워크 비효율을 완화하고 각 다운로드에 대해 적은 비용을 적용하고

그 다운로드를 써드하는 모든 피어에게 보상으로 해당 비용을 분배하여 파일 가용성을 높이는 인센티브 메커니즘으로 보상 시스템을 구현합니다. 이 비용은 지불 모델에 따라 컨텐츠의 총 비용에 보상 분배로 반영될 수 있으며, "무료" 컨텐츠의 경우 이 비용은 기존의 광고 또는 데이터 수집(사용자들이 오늘날 "무료" 컨텐츠에 지불하는 방식)을 대체할 수 있습니다. 각각의 다운로드가 보상의 원천이 되므로 시간이 지남에 따라 네트워크 참여자는 보상으로부터 훨씬 더 많이 벌 수 있고 그런 다음 선불로 다운로드 비용을 지불합니다.

### 2.3.2 파일 전달

파일 전달은 Ara의 콘텐츠 전송 네트워크를 유지하고 참여자가 보상을 받기 위한 주요 메커니즘입니다. Ara의 파일 전달 프로토콜은 4단계 핸드셰이크로 시작합니다.

1. 콘텐츠 요청자인 Alice는 일부 네트워크 탐색 프로토콜에서 콘텐츠 조각에 대한 다운로드 요청을 브로드캐스트합니다(CFSNet은 mDNS 및 BitTorrent를 포함한 여러 전략을 구현합니다).
2. 라이선스 검증자 및 콘텐츠 전달자인 Bob은 브로드캐스트를 수신하고 자신의 파일 가용성으로 응답합니다.
3. Alice는 응답 풀(swarm)에서 피어를 선택하고 자신의 DID와 함께 중계 공개 키로 메시지를 보냅니다.
4. Bob은 암호화된 방식으로 Alice의 메시지를 확인하고 그녀가 기본 AFS에 대한 라이선스를 구입했다는 것을 확인합니다.

이러한 매칭이 완료되면 파일 전송이 시작됩니다.

### 2.3.3 스마트 계약

Ara는 처음에 이더리움 메인넷에서 시작되었으며 이더리움 메인넷은 스마트 계약이 Ara 프로토콜군의 중심 구성요소 역할을 하게 되는 곳입니다. 이러한 스마트 계약은 DCDN, AraID 및 애플리케이션 계층 간의 상호운용성을 중재하고 촉진하며, 플랫폼의 모든 비 트랜지언트 속성과 엔티티가 다음을 포함한 이더리움 블록체인에 등록할 수 있게 합니다.

- 게시된 콘텐츠
- 구매
- 보상
- Ara 잔액

Ara의 스마트 계약 아키텍처는 보안과 수정 가능성을 염두에 두고 설계되었습니다 AFS의 판매 및 구매 방법, 보상 처리 및 배포 방법, 시스템 내에서 결제 처리 및 라우팅 방법의 진화하는 개념을 지원하기 위해 Ara는 게시된 각 AFS에 대해 프록시 계약을 배포합니다. 프록시는 AFS에 대한 비즈니스 로직을 정의하는 특정 버전의 AFS 표준과 연관된 레지스트리 계약을 통해 배치됩니다. 각 AFS에 대해 전체 AFS 표준을 배포하는 것은 비용이 많이 들고 AFS는 본질적으로 특정

표준에 고착되기에 업데이트하기가 어렵겠지만 프록시 아키텍처를 사용하면 AFS 수명 동안 단일 프록시를 배치하고 참조하는 AFS 표준 버전을 변경하여 업그레이드할 수 있습니다. 또한 프록시 아키텍처는 등록된 프록시 주소(즉, 유효한 AFS 콘텐츠)만을 라이브러리 계약의 사용자 라이브러리에 추가할 수 있도록 보장합니다.

### AFS 표준

AFS 표준은 AFS가 이더리움 블록체인에서 정의되고 구조화되고 독립적인 존재를 가질 수 있게 합니다. 여기에는 구입 및 보상 방법뿐 아니라 메타데이터 SLEEP 레지스터에서 트리 및 서명 파일을 저장하는 방법도 포함됩니다. 메타데이터 SLEEP 레지스터는 파일 이름, 크기 및 권한을 포함하여 AFS 내의 콘텐츠에 대한 메타데이터를 저장함과 동시에 콘텐츠 SLEEP 레지스터는 해당 파일의 원시 이진 콘텐츠를 저장합니다. 메타데이터 레지스터 내에서 트리 파일은 콘텐츠 레지스터에 있는 데이터를 구성하는 직렬화된 머클 트리를 나타내고 서명 파일은 직렬화된 트리의 서명된 루트를 저장합니다. CFS를 사용하면 이러한 파일을 디스크에 저장하게 되고 디스크에서 읽게 되지만 AFS를 사용하면 이러한 파일을 이더리움 블록체인에 쓰고 읽습니다. AFS는 자체 프록시를 통해 이러한 표준과 통신하므로 많은 수의 다양한 AFS 표준이 공존할 수 있으며 콘텐츠 생성자가 자신의 필요에 가장 적합한 표준을 선택할 수 있게 합니다.

프록시를 사용하면 AFS 표준이 스토리지에서 로직을 분리합니다. 여기에서 AFS 표준은 해당 버전의 표준을 사용하는 모든 AFS에 대한 로직 계층의 역할을 하고 각 프록시는 단일 AFS의 저장 계층의 역할을 합니다. 최소한 AFS 표준은 가격 책정, 구매, 보상 및 스토리지 기능의 구현을 실시하는 AFS 표준 추상 클래스를 구현해야 합니다.

가장 기본적이고 기본값인 AFS 표준의 경우 AFS 소유자만 가격을 수정할 수 있습니다. 구매 시, 이 가격은 구매자의 Ara 지갑에서 소유자의 Ara 지갑으로 이전됩니다. 기본 AFS 표준은 보상 예산에 대한 지원을 실시하며, 이러한 보상 예산은 다운로드 전에 제출되어야 합니다. 다운로드가 완료되면 참여하는 피어 사이의 예산이 할당되고 그런 다음에 피어는 보상을 교환받기 위해 필요한 잔액 유지를 철회하지 않았기에 부여된 보상을 교환할 수 있습니다.

콘텐츠 생성자의 재량에 따라 AFS 표준은 다음과 같이 많은 수의 맞춤형 상거래 제어를 지원할 수 있습니다:

1. **로열티**: 비율 내역별로 많은 수의 다양한 Ara 계정 간에 수익을 배분하도록 구매를 맞춤화할 수 있습니다.
2. **대량 구매**: 구매된 수량 기반으로 가격을 계층화할 수 있습니다.
3. **재판매 조건**: 구입한 콘텐츠는 콘텐츠 생성자가 지정한 최소 재판매 가격으로 여러 번 재판매할 수 있습니다.
4. **소유권 이전**: AFS 소유자는 또 다른 이더리움 주소로 소유권을 쉽게 이전할 수 있습니다.
5. **사전 주문**: 콘텐츠의 다운로드가 가능해지기 전에 콘텐츠를 구매할 수 있습니다. 구매자는 보상 예산을 미리 제출하여 AFS가 사용 가능하게 되는 즉시 다운로드를 시작할 수 있습니다.
6. **부록**: 콘텐츠 생성자는 AFS의 최대 판매 수를 정의할 수 있으며 그 후에는 AFS가 비공개되고 구입할 수 없게 됩니다.

이러한 상거래 제어는 모든 유형의 콘텐츠 생성자에게 자신의 정확한 사양에 대한 요구 사항에 맞게 맞춤화된 수의 모델과 비즈니스를 정의할 수 있는 기능을 제공합니다. 기존의 방법을 통해 구현하기 어려운 흥미로운 새 모델을 만들기 위해 여러 상거래 제어를 결합할 수 있습니다. 예를 들어, 음악을 리믹스하는 것을 즐기는 사람은 정의된 재판매 조건과 최소 재판매 가격과 함께 오리지널 아티스트의 트랙을 구매하고, 계속 오리지널 아티스트에게 구매 대금을 지불하면서 곡을 리믹스하고 리믹스한 버전을 판매할 수 있습니다.

이전에는 이러한 유형의 상거래 제어를 결합하려면 엄청난 양의 시간, 자본, 법적 개입이 필요했고 이는 소규모 콘텐츠 생성자에게는 심각한 진입 장벽이었습니다. 하지만 현재는 모든 사용자가 무료로 이용할 수 있습니다.

## 등록

프록시 아키텍처의 일부로써 레지스트리 계약은 두 가지 주요 기능을 제공합니다:

1. 프록시 공장 역할을 합니다.
2. 모든 AFS 표준 버전을 추적합니다.

AFS가 처음으로 게시되면 레지스트리는 해당 AFS에 대한 프록시를 배포하고, 해당 프록시와 지정된 AFS 표준 사이의 관계를 수립합니다. 프록시는 AFS 표준이 호출할 때마다 레지스트리에 각각의 AFS 표준의 주소를 문의하고 해당 주소로 호출을 위임합니다. 이 주소는 처리된 후에 프록시로 반환됩니다.

## 라이브러리

Ara 네트워크는 라이브러리 계약을 활용하여 구매 여부에 관계없이 사용자가 액세스할 수 있는 AFS에 대한 정규 사실 소스를 만듭니다. 콘텐츠를 구입할 때 AFS 표준에 있는 구매 기능은 라이브러리 계약에 있는 구매자의 라이브러리에 AFS DID를 자동으로 추가합니다. 이렇게 하면 사용자의 라이브러리에 대한 정보가 필요한 어떠한 서비스도 해당 정보에 대한 계약을 쿼리할 수 있습니다. 블록체인에 저장된 AFS DID를 사용하면 모든 서비스가 기본 콘텐츠로 해결할 수 있습니다. 라이브러리는 등록된 프록시만 사용자의 라이브러리에 해당 AFS를 추가하여 동의없이는 그 누구도 또 다른 사용자의 라이브러리를 조작할 수 없게 합니다.

### 3. 향후 개발

#### 3.1 모듈

Ara 플랫폼은 플랫폼 상단에서 실행할 수 있는 분산 서비스의 유형에 근본적으로 애그노스틱합니다. 모듈은 모듈 API를 구현하고 플랫폼 내에서 상호 교환하여 사용할 수 있는 분산형 및/또는 탈중앙형 서비스입니다. ERC-20 토큰 표준을 사용하여 이더리움의 모든 토큰을 다른 애플리케이션에서 재사용할 수 있는 방법과 마찬가지로 모듈 API를 사용하면 인터페이스를 구현하는 모든 분산 서비스를 플랫폼 전체에서 사용할 수 있습니다 [14]. 이는 Ara의 보상, 구매 및 지불 시스템을 활용할 수 있는 기능을 보유한 분산형 서비스의 생태계를 구축하는 데 전념하는 개발자 커뮤니티 형성을 용이하게 하며 본질적으로는 보상 가능한 분산 서비스 경제를 형성합니다. 보상 시스템을 활용할 방안을 찾는 모듈은 자체 보상 메커니즘 및 방법론을 정의하는 추가적인 스마트 계약 API를 구현할 것으로 예상됩니다. 각 모듈 스마트 계약은 분산 서비스를 사용하여 누적된 각각의 보상 배분을 저장하고 그에 따라 분배합니다.

#### 3.2 Ara 명칭 시스템(Ara Name System (ANS))

ANS는 도메인 명칭 시스템(Domain Name System (DNS))과 [9] 마찬가지로 Ara 네트워크 내에서 인증서를 등록, 쿼리, 호출 또는 철회하는 분산형 방식입니다. DNS가 인터넷의 애플리케이션 계층의 일부로 있는 반면(사람이 읽을 수 있는 URI를 기본 IP 주소로 해결하여 사용자 애이전트(예: 웹 브라우저)가 요청된 콘텐츠를 얻고 렌더링할 수 있음) ANS는 사람이 읽을 수 있는 이름을 DDO에 대한 궁극적인 해결을 위해 DID로 해결합니다. ANS는 두 번째 단계의 해결을 위해 후드 아래에 있는 ID 아카이버와 리졸버를 사용하여 DID에서 DDO를 제공합니다. ANS는 본질적으로 사람이 읽을 수 있는 URI에 대한 아

카이베이자 리졸버입니다. ANS의 예제 애플리케이션은 Ara 상단에 구축된 웹 브라우저의 컨텍스트에서 호스트 이름으로부터 DDO를 제공할 수 있습니다.

다양한 레코드 유형을 파악하기 위해 DNS가 자체 레코드를 분류하는 방법과 비슷하게 각 레코드는 TYPE 리소스 레코드를 저장합니다 [17]. TYPE 필드는 숫자 값으로 표시되며 이를 통해 나중에 ANS에 다른 유형의 레코드를 저장할 수 있습니다. 다음 표에서는 레코드 TYPE에 대해 설명합니다:

TYPE	Value	Description
USR	00	User
PCT	01	Published Content

ANS로 구성된 각 슈퍼노드 허브는 HyperDB 인스턴스를 실행합니다 [2]. HyperDB와 같은 분산형의 확장성이 매우 뛰어난 데이터베이스는 ANS와 같은 시스템에 적합할 수 있도록 하는 몇 가지 기능을 제공합니다. 첫 번째는 트라이의 HyperDB 사용입니다. 각 노드가 자식 노드에 대해 접두어인 검색 트리입니다 이름을 트라이와 함께 저장하여 데이터베이스에서 수천 개의 항목을 사용해도 검색이 저렴하고 빠르다는 것을 보장할 수 있습니다. 트라이에서 검색은  $O(n)$ 이며, 여기서  $n$ 은 검색 중인 키의 길이입니다. 또한 HyperDB는 분산 시스템 내의 이벤트의 원인을 추적하는 벡터 시계를 사용하여 노드가 동기화되지 않는 경우를 방지합니다 [10].

### 4. 감사의 말

이 백서는 Littlstar 및 Token Foundry의 지원을 통해 만들 수 있었습니다. 기여해주신 Logan Dwight, Andrew Grathwohl, Brandon Plaster에게 특별한 감사를 전합니다.

## 약자

**AFS** Ara File System. 1, 3–5, 9

**ANS** Ara Name System. 1, 10

**CFS** Conflict-Free File System. 4, 5

**CFSNet** Conflict-Free File System Network. 4

**DCDN** Decentralized Content Delivery Network. 3–5, 8

**DDO** DID Descriptor Object. 4, 5, 10

**DID** Decentralized Identifier. 4, 5, 9, 10

**DNS** Domain Name System. 4, 10

**DPKI** Decentralized Public Key Infrastructure. 4

**FHS** Filesystem Hierarchy Standard. 4, 5

**PII** Personally-Identifiable Information. 4

**SLEEP** Syncable Ledger of Exact Events Protocol. 4

## 참고 문헌

- [1] Code for Science Buus, Ogden. Sleep - syncable ledger of exact events protocol. <https://github.com/datproject/docs/blob/master/papers/sleep.pdf>, Aug 2017.
- [2] Mathias Buus. Hyperdb. <https://github.com/mafintosh/hyperdb>, Aug 2017.
- [3] Cisco. Cisco visual networking index predicts global annual ip traffic to exceed three zettabytes by 2021. <https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1853168>, Jun 2017.
- [4] Stewart Clarke. Piracy set to cost streaming players more than \$50 billion, study says. <http://variety.com/2017/tv/news/piracy-cost-streaming-players-over-50-billion-1202602184/>, Oct 2017.
- [5] Palatinus et al. Mnemonic code for generating deterministic keys. <https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki>, Sept 2013.
- [6] Russell Hardin. The free rider problem. <https://plato.stanford.edu/entries/free-rider>, May 2003.
- [7] Ahamad Jun. Incentives in bittorrent induce free riding. [https://disco.ethz.ch/courses/ws0506/seminar/papers/freeriding\\_incentives.pdf](https://disco.ethz.ch/courses/ws0506/seminar/papers/freeriding_incentives.pdf), Aug 2005.
- [8] The Linux Foundation LSB Workgroup. Filesystem hierarchy standard. [https://refspecs.linuxfoundation.org/FHS\\_3.0/fhs-3.0.pdf](https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf), Mar 2015.

- [9] Paul Mockapetris. Domain names - implementation and specification. <https://tools.ietf.org/html/rfc1035>, Nov 1987.
- [10] Multiple. Vector clock. [https://en.wikipedia.org/wiki/Vector\\_clock](https://en.wikipedia.org/wiki/Vector_clock).
- [11] Markus Sabadello. A universal resolver for self-sovereign identifiers. <https://medium.com/decentralized-identity/a-universal-resolver-for-self-sovereign-identifiers-48e6b4a5cc3c>, Nov 2017. Accessed on 2018-04-24.
- [12] Stephen E. Siwek. The true cost of sound recording piracy in the us economy. [https://www.riaa.com/wp-content/uploads/2015/09/20120515\\_SoundRecordingPiracy.pdf](https://www.riaa.com/wp-content/uploads/2015/09/20120515_SoundRecordingPiracy.pdf), Aug 2007.
- [13] Rebooting the Web-of Trust. Decentralized public key infrastructure. <http://www.weboftrust.info/downloads/dpki.pdf>, Dec 2015. Accessed on 2018-04-19.
- [14] Buterin Vogelsteller. Erc-20 token standard. <https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md>, Nov 2015.
- [15] W3C. Decentralized identifiers. <https://w3c-ccg.github.io/did-spec>, Apr 2018. Accessed on 2018-05-01.
- [16] Music Business Worldwide. Why does the riaa hate torrent sites so much? <https://www.musicbusinessworldwide.com/why-does-the-riaa-hate-torrent-sites-so-much/>, Dec 2014.
- [17] Inc ZyTrax. Dns resource records (rrs). <http://www.zytrax.com/books/dns/ch8/>, Oct 2015.

# 부록

## I. 완전히 성장한 Ara 플랫폼 토큰 경제(초안)

### 개요

기존의 클라우드 서비스는 사내 인프라의 구매 및 관리에서 융통성, 민첩성, 비용 절감을 제공하여 명성을 높여 왔습니다. 많은 클라우드 서비스가 장기간의 책무와 계약이 포함된 복잡하고 모호한 것으로 악명 높은 가격 모델을 사용하며 종종 고객마다 개별적으로 협상합니다. 그리고 이러한 모델은 처음부터 제공할 수 있는 유연성과 민첩성을 직접적으로 약화시킵니다. [1] 최근 몇 년 동안, P2P CDN은 새롭게 등장한 SaaS의 동영상 전송을 위한 하이브리드 호객 솔루션으로 주목을 받기 시작했습니다. 이러한 SaaS는 동영상 시청자의 기기를 활용하여 더 단순한 가격 모델로 확장성이 매우 뛰어난 솔루션을 구축하면서도 중앙집중화의 여러 소스를 유지합니다. 즉, 비즈니스 모델과 “2급 시민”으로서의 사용자의 기여를 통해 이를 구현하며 여기에서 사용자 기기의 활용은 그들의 동의나 재정적 보상 없이 발생합니다. Ara는 네트워크 참여자들에게 자신의 기기 사용에 대한 보상을 해줌으로써 이를 한 단계 더 끌어 올립니다.

기존의 고전적인 클라우드 인프라 비용을 대체하기 위해 Ara 플랫폼은 원래의 프로토콜 유ти리티 토큰인 Ara 토큰을 구현합니다. 이 토큰은 Ara 플랫폼에 사용되어 건강하고 정직한 네트워크 행동을 위한 암호 경제적 인센티브를 창출하고 콘텐츠 소비자와 생성자 사이의 직접적인 참여를 더욱 활성화할 뿐만 아니라 플랫폼이 채택되도록 장려할 수 있습니다. Ara 토큰은 네트워크가 제공하는 가치 구성원의 캡슐화로 볼 수 있으며 보상된 각 토큰은 네트워크 유ти리티에서의 미미한 증가를 나타냅니다.

### Ara 토큰

모듈로 알려진 Ara의 SDK를 사용하여 구축된 분산 서비스는 모두 Ara 네트워크에서 구입, 판매, 요청 및 이행할 수 있습니다. 모듈 작업은 네트워크에 있는 피어에게 아웃소싱되고 성공적으로 완료하면 해당 피어는 Ara 토큰으로 보상받을 수 있습니다. 개방적이고 경쟁적인 시장을 촉진하기 위해 Ara는 서비스 요청자가 요청한 서비스에 대해 보상 분배 또는 보상금을 정의할 수 있게 하고 서비스 제공자는 최소 수용 보상금을 정의할 수 있게 합니다. 인간의 지능이 필요한 작업을 크라우드소싱하기 위한 시장인 Amazon Mechanical Turk와 마찬가지로 Ara는 분산 컴퓨팅 또는 네트워킹 작업을 아웃소싱하기 위한 시장을 만듭니다. 모듈은 분산형 트랜스코딩과 같이 일회성 온디맨드 작업이거나 P2P 멀티플레이어 게임 서버와 같이 지속적인 반복 서비스일 수 있습니다.

## 기능

Ara 토큰은 다양한 방법으로 네트워크 전체에서 사용할 수 있습니다.

- 소비자의 경우, Ara 토큰을 사용하여 즐거움을 위한 디지털 콘텐츠에서부터 참여할 새로운 모듈에 이르기까지 어떠한 종류의 구매라도 할 수 있습니다.
- 서비스 요청자는 Ara 토큰을 사용하여 작업 요청을 시작하고 해당 작업을 성공적으로 완료하도록 보상금을 설정할 수 있습니다.
- 서비스 제공자는 보상에 대한 대가로 작업을 이행하겠다는 약속으로써 Ara 토큰을 보증금으로 낼 수 있습니다. (보증금은 요청 시 돌려줄 수 있지만 해당 공급자는 더 이상 보상을 못 받게 됩니다.)
- 개발자는 Ara 토큰을 사용하여 새 모듈을 네트워크에 배포할 수 있습니다.

이러한 각각의 역할이 심하게 겹치기 때문에 개발자가 보상으로 얻은 Ara 토큰을 모듈을 구입하는 데 사용하여 새로운 작업 요청을 시작할 수 있는 것과 같이 서비스 제공자는 보상으로 얻은 동일한 Ara 토큰을 새로운 콘텐츠를 구입하는 작업에 사용할 수 있습니다.

## 시장 역학

네트워크 구성원이 참여하는 작업 또는 서비스를 결정하는 데 전적인 권한을 가지고 있기 때문에 네트워크는 경제적 균형이 발생하는 자유 시장을 형성합니다. 각 모듈은 특정 모듈의 작업에 대한 요청자 및 공급자의 행동에 의해 지배되는 자체적인 경제를 가질 수 있습니다. 예를 들어, 분산형 동영상 트랜스코드는 동영상 제작자에게 특성상 시급한 일이 될 수 있으며 그 결과로 분산형 동영상 트랜스코드에 대한 수요 가격이 비탄력적으로 될 수 있습니다. (즉, 동영상 제작자는 서비스 공급자에게 얼마만큼 보상을 해야 하는지에 대해 상대적으로 무심합니다.) 그러므로 판매자의 시장은 분산형 트랜스코드 서비스 공급자가 보상 배분을 결정하기가 유리하기에 보상금이 오르는 시장에서 형성됩니다. 이와 비슷하게 P2P 게임 서버 모듈은 매우 많이 요청될 수 있지만 상대적으로 소수의 서비스 공급자를 가집니다. 다시 말하자면, 판매자의 시장이 형성되고 보상금은 오릅니다. 반면에 분산된 멀티 플랫폼 모듈은 소수의 사람들에 의해서만 요청될 수 있지만 자격을 갖춘 많은 서비스 공급자를 가집니다. 상대적으로 수요가 낮기 때문에 많은 공급자들이 최소 보상금 요건을 너무 높게 설정한 경우에는 기회를 놓치게 됩니다. 구매자의 시장이 형성되고 보상금이 낮아집니다.

보상금 설정하기는 모듈에 대한 요건이 아니며 보상금 설정 방법에 대한 표준화된 모델은 없다는 점을 유의하는 것이 중요합니다. 목표는 서비스 요청자와 서비스 공급자의 인센티브를 조정하고 모든 유형의 인센티브 모델을 지원할 뿐만 아니라 전 세계의 인프라 및 네트워킹 기능에 대한 다양한 고정 비용을 수용하는 것입니다.

## 인센티브 구조

이 모델이 서비스 요청자와 서비스 공급자 사이의 인센티브 조정을 어떻게 지원하는지 더 잘 이해하기 위해 양쪽 당사자 모두의 경제적 이익에 대한 일반적인 설명을 해드리겠습니다. 서비스 요청자는 가장 낮은 비용으로 요청을 수행하기를 원합니다. 반면에 서비스 제공자는 가장 많이 차별하는 서비스의 수가 최대가 되도록 최적화하기를 원합니다. 다시 말해, 요청자와 공급자 모두가 가격에 동의하는 한 네트워크 유트리티를 최대화하는 것이 양쪽 모두에게 최선의 이익이 됩니다. 따라서, 보상금과 작업의 균형을

가장 잘 잡는 서비스가 수행될 가능성이 가장 높으며 효율성을 향상시키기 위해 분산 서비스 설계에서 경쟁력 있는 보상금과 혁신을 모두 촉진하는 시장 압력이 생깁니다.

플랫폼에서 실행할 수 있는 다양한 분산 서비스는 작업 단위의 보상금(*UWR*, 보상금이 분리되고 보상되는 입증 가능한 작업의 기본 단위)과 보상금 모델(보상금 지급 방법을 관리하는 조건)을 결정할 때 유연성이 필요합니다. P2P 멀티플레이어 게임 서버는 *UWR*의 역할이 되는 요청의 수를 사용할 수 있으며 해당 서버 모듈을 호출하는 게임 개발자는 구독 기반의 반복적인 보상금 모델이 가장 적합하다고 판단할 수 있습니다. 반면에 분산형 트랜스코딩 서비스는 분당 트랜스코드된 바이트의 수를 *UWR*로 사용할 수 있으며 동영상 제작자는 트랜스코드당 보상금을 지불할 수 있습니다.

서비스 공급자는 참여하고 보상을 받기 위해 Ara 토큰을 걸 수 있습니다. 보상금과 마찬가지로 서비스 요청자는 최소한의 자분을 결정할 수 있으며 해당하는 경우 해당 공급자는 서비스에 참여하기 위해 보증금을 내야 합니다. 최소 자분 가치는 서비스가 요구하는 약정 수준을 나타내야 하며 일단 서비스가 성공적으로 완료되면 보상금과 함께 반환됩니다. 또한 *UWR*을 결정 과정의 일부로써 서비스는 이행을 확인하기 위한 증거를 정의해야 합니다.

또는 서비스 공급자가 리소스를 서비스 전용으로 사용하기 위해 구독료를 설정할 수 있습니다. 이러한 전용 공급자는 슈퍼노드로 알려져 있으며 그들의 자분은 구독이 종료될 때까지 스마트 계약에 예탁됩니다. 슈퍼노드는 비 전용 상대방보다 더 신뢰할 수 있는 경향이 있기 때문에 그들이 구독료 설정을 어떻게 하느냐에 따라 시장 역학을 제어할 수 있습니다. 보고타의 슈퍼노드는 하드웨어 및 인터넷 비용이 더 많이 들기 때문에 로스 엔젤레스의 슈퍼노드보다 더 많이 청구할 수 있습니다.

## 네트워크 효과

새로운 콘텐츠를 네트워크에 도입하는 것은 콘텐츠 중복 및 가용성 전용에 대한 Ara 노드인 DCDN 슈퍼노드의 호출을 포함합니다. 단일 파일에 대한 보상금이 일정하다고 가정하면 DCDN 및 고정 인센티브를 가지는 모든 통화 기반의 P2P 파일 공유 시스템 [2]에 있는 모든 피어의 해당 파일에서 받는 잠재적 보상에 대해 해당 파일을 공유하는 단일 피어로부터의 파일 가용성이 미미하게 증가하는 효과는 서브모듈러 집합 함수를 사용하여 모델링할 수 있으며, 이는 수익 감소를 설명하는 함수로 직관적으로 생각할 수 있습니다. 이러한 속성으로 인해 DCDN 슈퍼노드는 구독 현상금 모델을 배포하여 가용성이 증가함에 따라 콘텐츠를 호스팅하는 기회비용이 증가하는 것에 대응할 수 있습니다.

콘텐츠 게시자는 콘텐츠별로 선택한 지리적 위치에서 최대한 많은 또는 적은 수의 슈퍼노드를 호출하거나 구독할 수 있습니다. 그런 후에 게시자에게는 전 세계에서 해당 콘텐츠를 어느 정도 쉽게 사용할 수 있게 할지 자유롭게 결정할 자유가 있습니다. 이를 통해 전 세계의 이용 가능한 모든 슈퍼노드를 호출하여 전 세계 잠재 고객을 지원하고자 하는 대형 미디어 배급자를 지원할 수 있으며, 주요 잠재 고객을 주로 유럽인으로 식별하여 유럽 슈퍼노드의 우선순위를 결정하는 인디 콘텐츠 생성자를 지원할 수도 있습니다. 또한 콘텐츠 게시자는 각 콘텐츠 다운로드에 대한 보상 배분을 결정합니다. 따라서 원하는 수준의 참여를 달성하기 위한 최적의 보상 배분 및 슈퍼노드 배포가 존재합니다(예: 파일 가용성).

## 참고 문헌

- [1] Enterprise Strategy Group (2015, June), *Price Comparison: Google Cloud Platform vs. Amazon Web Services*, <https://cloud.google.com/files/esg-whitepaper.pdf>
- [2] M. Salek, S. Shayandeh, and D. Kempe, *You Share, I Share: Network Effects and Economic Incentives in P2P File-Sharing Systems* <https://arxiv.org/pdf/1107.5559.pdf>

## II. DCDN 비용 분석 by Lester Kim

### 소개

잠재적인 파트너에 대한 스트리밍 비용을 계산하기 위해서는 단위 시간  $T$ (초 단위) 당 소비자에게 얼마나 많은 데이터  $B$ (바이트 단위)를 제공해야 하는지 알아야 합니다.  $N \in \mathbb{N}$ 인 경우,  $N$ 개의 업로더 노드 그룹을 가정해 보겠습니다.  $\forall n \in \{1, \dots, N\}$ 이면 그룹  $n$ 의 평균 대역폭은  $b_n$ (노드 당 바이트/초 단위)입니다.  $q_n$ 을 그룹  $n$  노드의 수량이라고 하겠습니다.  $\mathbf{b} = [b_1 \dots b_N]^\top$  및  $\mathbf{q} = [q_1 \dots q_N]^\top$ 라고 하겠습니다. 따라서  $\frac{B}{T}$ 에 의해 제약된 초당 전달되는 바이트의 양은 다음과 같습니다.

$$g(\mathbf{q}) = \mathbf{b} \cdot \mathbf{q} = \frac{B}{T}. \quad (1)$$

분포  $C(\mathbf{q})$ 의 비용을 최소화하기 위해 최적의  $\mathbf{q}^*$ 를 찾고자 합니다.  $\mathbf{p} = [p_1 \dots p_N]^\top$ 이고  $p_n$ 이 그룹  $n$ 에 대한 노드당 가격인 경우라면 다음과 같습니다.

$$C(\mathbf{q}) = \mathbf{p} \cdot \mathbf{q}. \quad (2)$$

### 업로더의 수익 최대화

$\mathbf{p}$ 를 결정하기 위해 수익을 극대화하는 기업의 행위를 살펴보겠습니다.  $f$ 를 에너지 입력  $E$ (kWh 단위) 및 출력  $q$ (노드 단위)가 있는 생산 함수라고 하겠습니다. 이 생산 함수를 다음과 같이 모델링합니다.

$$f(E) = AE^\alpha \quad (3)$$

여기에서  $A$ 는 생산 요소(노드/kWh $^\alpha$ )이고  $\alpha \in [0, 1]$ 은 생산의 탄력성(입력 증가율 대비 출력 증가율)입니다 [1].

노드가 데이터 업로드를 시작할 때의 전력 증가를  $P$ (kWh/s 단위)라고 하겠습니다. 여기에는 네트워크 인터페이스 컨트롤러(NIC)를 통해 데이터를 전송하는 것이 포함되지만 (꺼진 상태 또는 대기 모드에서) 기기를 켜는 것도 포함될 수 있습니다. 각 노드의 전력이  $P$ 이면 일부  $E$ 에 대해서는 단일 노드가  $\frac{E}{P}$ 초 동안 실행될 수 있습니다. 그러나 작업을 완료하기 위한 시간 제약 조건  $T$ 가 주어지면  $\frac{E}{PT}$  노드가 있어야 합니다. 그러므로 다음과 같습니다.

$$A = \frac{D}{(PT)^\alpha} \quad (4)$$

여기에서  $D$ 는 전체 요인 생산성(노드 단위)입니다 [2].

$p$ 는 노드의 가격이고  $p_E$ 는 에너지의 가격(kWh 당)입니다. 회사의 수익 함수  $\pi$ 는 다음과 같습니다

$$\pi(q, E) = pq - p_E E. \quad (5)$$

대역폭 비용은 몇 개월<sup>1</sup>이 아닌 몇 초를 보는 짧은 기간에서의 고정 비용이기 때문에 무시합니다.

적어도 최소의 출력 요건  $q$ 와 다음과 같은 조건이 주어졌을 때 기업의 수익을 극대화하고자 합니다.

$$\max_{q, E} \pi(q, E) \quad \text{s.t. } f(E) \geq q. \quad (6)$$

이 문제를 풀기 위해 다음과 같은 라그랑지안을 도입하겠습니다.

$$\mathcal{L}(q, E, \lambda) = pq - p_E E - \lambda(AE^\alpha - q). \quad (7)$$

편미분을 취하고 0으로 놓으면 다음과 같습니다.

$$\frac{\partial \mathcal{L}}{\partial q} = p + \lambda = 0 \quad (8)$$

$$\frac{\partial \mathcal{L}}{\partial E} = -p_E - \lambda A\alpha E^{\alpha-1} = 0 \quad (9)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = q - AE^\alpha = 0. \quad (10)$$

이 1차 미분 방정식을 풀면 다음과 같습니다.

$$q^* = \left( \frac{\alpha A^{\frac{1}{\alpha}} p}{p_E} \right)^{\frac{1}{1-\alpha}}. \quad (11)$$

이것을 다시 쓰면( $q$ 에서 별표 삭제) 다음과 같습니다.

$$p = \frac{p_E}{\alpha} \left( \frac{q^{1-\alpha}}{A} \right)^{\frac{1}{\alpha}}. \quad (12)$$

이 수식을 사용하면 원하는 노드의 수를 얻기 위해  $p$ 가 무엇이 되어야 하는지를 생성자가 알 수 있습니다.

(12)에서  $q$ 에 관한 최적의 수익은 다음과 같습니다.

$$pq = \frac{p_E}{\alpha} \left( \frac{q}{A} \right)^{\frac{1}{\alpha}}. \quad (13)$$

---

<sup>1</sup>대역폭이 포함되어 있다 하더라도 초당 비용은 에너지의 비용과 같은 크기의 차원을 가집니다. 뉴욕시에서는 50Mbps에 \$3 x 10<sup>-5</sup>/초의 비용이 듭니다 [3].

## 배포자의 비용 최소화

회사의 수익이 소비자(생성자)에게 지출되고 있으므로 (2)를 다음과 같이 쓸 수 있습니다.

$$C(\mathbf{q}) = \frac{p_E}{\alpha} \sum_{n=1}^N \left( \frac{q_n}{A_n} \right)^{\frac{1}{\alpha}}. \quad (14)$$

생성자의 비용 최소화 문제는 다음과 같습니다.

$$\min_{\mathbf{q}} C(\mathbf{q}) \quad \text{s.t. } g(\mathbf{q}) \geq \frac{B}{T}. \quad (15)$$

라그랑지안은 다음과 같습니다.

$$\mathcal{L}(\mathbf{q}, \lambda) = C(\mathbf{q}) - \lambda(g(\mathbf{q}) - \frac{B}{T}). \quad (16)$$

1차 편미분 방정식은 다음과 같습니다.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{q}} = \frac{\partial C}{\partial \mathbf{q}} - \lambda \frac{\partial g}{\partial \mathbf{q}} = 0 \quad (17)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \frac{B}{T} - g(\mathbf{q}) = 0. \quad (18)$$

(14), (4), (1)으로부터 다음과 같이 쓸 수 있습니다.

$$\frac{\partial C}{\partial \mathbf{q}} = \frac{p_E T \mathbf{P}}{\alpha^2} \circ (\mathbf{q}^{\circ(1-\alpha)} \oslash \mathbf{D})^{\circ\frac{1}{\alpha}} \quad (19)$$

$$\frac{\partial g}{\partial \mathbf{q}} = \mathbf{b} \quad (20)$$

$(\mathbf{D}, \mathbf{P}) = ([D_1 \dots D_N]^\top, [P_1 \dots P_N]^\top)$ 인 경우, " $\circ$ ", " $\circ\circ$ ", " $\oslash$ "은 각각 아다마르(entrywise) 곱, 지수 및 나누기입니다 [4]. 따라서  $\forall m, n \in \{1, \dots, N\}$ 이면,

$$\frac{P_m^\alpha q_m^{1-\alpha}}{b_m^\alpha D_m} = \frac{P_n^\alpha q_n^{1-\alpha}}{b_n^\alpha D_n}. \quad (21)$$

따라서

$$b_m q_m = \left( \frac{b_m D_m P_n^\alpha}{P_m^\alpha b_n D_n} \right)^{\frac{1}{1-\alpha}} b_n q_n. \quad (22)$$

(22)를 (18)과 결합하면

$$\mathbf{q}^* = \frac{B\mathbf{b}^{\circ-1}}{T\kappa} \circ (\mathbf{b} \circ \mathbf{D} \oslash \mathbf{P}^{\circ\alpha})^{\circ\frac{1}{1-\alpha}} \quad (23)$$

$$C^* = \frac{p_E T}{\alpha} \left( \frac{B}{T\kappa^{1-\alpha}} \right)^{\frac{1}{\alpha}} \quad (24)$$

여기서

$$\kappa \equiv \sum_{m=1}^N \left( \frac{b_m D_m}{P_m^\alpha} \right)^{\frac{1}{1-\alpha}}. \quad (25)$$

**사례:**  $\alpha = 1$

$\alpha = 1$ 인 경우, (23) 및 (24)는 다음과 같이 됩니다.

$$q_n^* = \begin{cases} \frac{B}{|\Upsilon|Tb_n} & n \in \Upsilon \\ 0 & n \notin \Upsilon \end{cases} \quad (26)$$

$$C^* = \frac{p_E B P_n}{b_n D_n} \quad \text{any } n \in \Upsilon \quad (27)$$

여기서

$$\Upsilon \equiv \left\{ n \in \{1, \dots, N\} \mid n = \arg \max_{1 \leq m \leq N} \frac{b_m D_m}{P_m} \right\}. \quad (28)$$

$\forall n \in \Upsilon$ 인 경우 그룹  $n$ 의 각 노드는 데이터의  $\frac{B}{|\Upsilon|q_n^*}$  ( $= b_n T$ )를 전달하고 보상으로 최소한  $\frac{p_E P_n T}{D_n}$ 을 받습니다. 그러나  $\mathbf{q}^*$ 에 대한 여러 개의 답이 있습니다. 예를 들어 임의의  $n \in \Upsilon$ 에 대해서 그룹  $n$ 은  $\frac{B}{Tb_n}$  노드를 사용하여 모든 작업을 수행할 수 있습니다.

## 예제

다음과 같을 때

$$\alpha = 1 \quad (29)$$

$$B = 1 \text{ GB} \quad (30)$$

$$N = 2 \quad (31)$$

$$p_E = \$0.2321/\text{kWh} [5] \quad (32)$$

$$T = 1 \text{ s} \quad (33)$$

$$\mathbf{b} = \begin{bmatrix} 100 \text{ MB/s} \\ 1 \text{ MB/s} \end{bmatrix} [6] \quad (34)$$

$$\mathbf{D} = \begin{bmatrix} 1 \text{ node} \\ 1 \text{ node} \end{bmatrix} \quad (35)$$

$$\mathbf{P} = \begin{bmatrix} 200 \text{ W} \\ 2 \text{ W} \end{bmatrix} [7][8] \quad (36)$$

생성자에 대한  $\mathbf{q}^*$  및  $C^*$  의 예를 찾기 위해 뉴욕시의 예를 들어 보겠습니다. 그러면,

$$\mathbf{q}^* = \begin{bmatrix} 5 \text{ nodes} \\ 500 \text{ nodes} \end{bmatrix} \quad (37)$$

$$C^* \approx \$1.29 \times 10^{-4}. \quad (38)$$

이것은 AWS Cloudfront의 온디맨드 가격(\$0.020/GB - \$0.085/GB)보다 155,659배 (99.35% - 99.85%) 저렴합니다 [9]. 그룹 1의 각 노드는 100MB를 처리하게 되지만 그룹 2의 각 노드는 1MB를 처리하게 됩니다. 그룹 1과 그룹 2의 각 노드는 각각  $\$1.29 \times 10^{-5}$  and  $\$1.29 \times 10^{-7}$ 보다 더 많이 필요하게 됩니다.

이것을 균형 있는 시각으로 보기 위해 Netflix를 잠재적인 파트너로 생각해 보십시오. 2017년에 Netflix의 콘텐츠는 하루 평균 140억 시간 이상 시청되었습니다 [10]. 평균적으로 Netflix의 동영상은 1GB/시간입니다 [11]. Ara 플랫폼에서 51.1 [12] 엑사바이트의 연간 지출은 660만 달러 (초당 0.2106달러)에 불과합니다. Netflix의 스트리밍 비용은 \$0.03/GB로 [13] 추정하면 15억 달러/1년(\$46.61/초)이 됩니다. Ara 네트워크를 사용하면 Netflix의 2017년 순이익인 5억 5,890만 달러의 거의 네 배가 됩니다 [14]. (맨해튼에는 166만 [15] 명이 살고 있고 그 중 Netflix 사용자는<sup>2</sup> 287,008명으로 321.45TB/일, 3.72GB/s를 스트리밍합니다. 이러한 양을 소화하려면 41.47달러/일의 속도로 3,721 개의 그룹 2 노드가 필요합니다.

---

<sup>2</sup>2018년 1분기 말에, Netflix는 5,671만 명의 미국 가입자와 1억 2천 5백만 [16] 명의 전 세계 가입자를 보유했습니다. 미국에는 3억 2,800만 [17] 명이 있으므로 비례적으로 맨해튼에는 287,008명의 Netflix 가입자가 있습니다.

## 참고 문헌

- [1] Wikipedia (2018, April 22), *Cobb-Douglas production function*, [https://en.wikipedia.org/wiki/Cobb-Douglas\\_production\\_function](https://en.wikipedia.org/wiki/Cobb-Douglas_production_function)
- [2] Wikipedia (2018, June 5), *Total factor productivity*, [https://en.wikipedia.org/wiki/Total\\_factor\\_productivity](https://en.wikipedia.org/wiki/Total_factor_productivity)
- [3] Spectrum (2017, December 29), *Broadband Label Disclosure*, p.2, [https://www.spectrum.com/content/dam/spectrum/residential/en/pdfs/policies/Broadband\\_Label\\_Disclosure\\_Charter\\_122917.pdf](https://www.spectrum.com/content/dam/spectrum/residential/en/pdfs/policies/Broadband_Label_Disclosure_Charter_122917.pdf)
- [4] Wikipedia (2018, March 10), *Hadamard product (matrices)*, [https://en.wikipedia.org/wiki/Hadamard\\_product\\_\(matrices\)](https://en.wikipedia.org/wiki/Hadamard_product_(matrices))
- [5] Electricity Local (2018, June 19), <https://www.electricitylocal.com/states/new-york/new-york>
- [6] Wikipedia (2018, June 19), *Bandwidth (computing)*, [https://en.wikipedia.org/wiki/Bandwidth\\_\(computing\)](https://en.wikipedia.org/wiki/Bandwidth_(computing))
- [7] Energuide.be (2018, June 19), *How much power does a computer use? And how much CO<sub>2</sub> does that represent?*, <https://www.energuide.be/en/questions-answers/how-much-power-does-a-computer-use-and-how-much-co2-does-that-represent/54/>
- [8] R. Sohan, A. Rice, A. W. Moore, and K. Mansley, "Characterizing 10 Gbps Network Interface Energy Consumption," *The 35th Annual IEEE Conference on Local Computer Networks (LCN) Short Papers*, University of Cambridge, Computer Laboratory, July 2010, <https://www.cl.cam.ac.uk/acr31/pubs/sohan-10gbpower.pdf>.
- [9] Amazon Web Services Pricing (2018, June 19), *Amazon Cloudfront Pricing*, <https://aws.amazon.com/cloudfront/pricing>
- [10] L. Matney (2017, December 11), "Netflix users collectively watched 1 billion hours of content per week in 2017," *Techcrunch*, <https://techcrunch.com/2017/12/11/netflix-users-collectively-watched-1-billion-hours-of-content-per-week-in-2017>
- [11] K. Hubby (2017, May 23), "The surprising amount of data Netflix uses," *The Daily Dot*, <https://www.dailydot.com/debug/how-much-data-netflix-use/>
- [12] Wikipedia (2018, June 20), *Exabyte*, <https://en.wikipedia.org/wiki/Exabyte>
- [13] D. Rayburn (2009, July), "Stream This!: Netflix's Streaming Costs," *Streaming Media* (June/July 2009), <http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/Stream-This!-Netflixs-Streaming-Costs-65503.aspx>
- [14] Netflix (2018, January 1), p.40, <https://ir.netflix.com/static-files/20c3228d-bf1f-4956-a169-c8b76911ecd5>
- [15] Wikipedia (2018, July 5), *Manhattan*, <https://en.wikipedia.org/wiki/Manhattan>

- [16] Statistica (2018, July 5), *Number of Netflix streaming subscribers in the United States from 3rd quarter 2011 to 1st quarter 2018 (in millions)*, <https://www.statista.com/statistics/250937/quarterly-number-of-netflix-streaming-subscribers-in-the-us/>
- [17] United States Census Bureau (2018, July 5), *U.S. and World Population Clock*, <https://www.census.gov/popclock/>

### III. 예상되는 Ara 보상 분석 by Lester Kim

#### 네트워크 모델

$V$ 는 노드를 나타내는 각각의 꼭지점이 있는  $N$  꼭지점을 포함하고,  $E$ 는 두 노드 사이의 통신 채널을 나타내는 각각의 모서리가 있는  $\frac{N(N-1)}{2}$  개의 모서리를 포함하는 경우, Ara 네트워크의 노드를 [1]  $G = (V, E)$ 로 완전한 그래프로 표시해 보겠습니다.  $C$ 를 모든 노드가 갖고 싶어하는 콘텐츠(우리의 경우 디지털 엔터테인먼트 파일 집합)의 일부 모음이라 하고, 부분 집합  $S \subseteq V$ 에  $C$ 가 있는 모든 노드가 포함된다고 하겠습니다(즉,  $S = \{v \in V : C \in v\}$ ).

시간을  $t \in \mathbb{N}$ 이라고 하겠습니다.  $t = 0$ 에서  $|S| = 1$ 이므로 콘텐츠  $C$ 를 갖는  $v_0 \in V$ 가 하나만 존재합니다. 그러므로  $C$ 의 사본을 네트워크의 다른 노드에 전달할 수 있는 꼭지점은 단 한 개입니다. 다른 모든  $N - 1$  노드는  $C$ 를 원하고  $v_0$ 에는 단 하나의 노드에만  $C$ 를 전달하기에 충분한 대역폭을 가지고 있다고 가정하겠습니다.  $t = 0$ 에서  $t = 1$ 까지  $|S|$ 은 1에서 2로 증가합니다. 일반적으로는 시간  $t$ 에서는 다음과 같습니다.

$$|S| = \begin{cases} 2^t & 0 \leq t < \log_2 N \\ N & t \geq \log_2 N. \end{cases} \quad (39)$$

$|S| = N$   $t = \lceil \log_2 N \rceil$ 에서 시작하는 점을 유의해 주십시오.

$\forall s \in S, s$ 는  $v$ 가  $s$ 에게  $p$ 를 지불하는 경우에만 일부  $v \in V \setminus S$ 을  $C$ 에 전달합니다.  $M$ 을 엔터테인먼트 전달을 위한 네트워크의 총 예산이라고 합시다. 이것을  $N$  노드로 고르게 나누면  $p = M/N$ 이 됩니다.

$t = 0$ 에서 유일한  $v_0 \in S$ 는 일부  $v_1 \in V \setminus S$ 로부터  $p$ 를 받습니다. 그런 다음  $t = 1$ 에서  $v_0, v_1 \in S$ 는 일부  $v_2, v_3 \in V \setminus S$ 로부터  $p$ 를 각각 받습니다. 임의의  $t < \lfloor \log_2 N \rfloor$ 에서,  $S$ 에 있는 각각의  $|S| = 2^t$  노드는  $V \setminus S$ 에 있는  $2^t$  노드로부터  $p$ 를 받습니다.  $t = \lfloor \log_2 N \rfloor$ 에서  $|S| > \frac{N}{2}$ 이므로  $C$ 의 수요자보다 더 많은 공급자가 있습니다. 이런 상황이 발생하면  $S$ 에서  $N - 2^t$ 개의 노드가 무작위로 선택되어  $C$ 를 전달합니다.  $t = \lceil \log_2 N \rceil$ ,  $S = V$ 에서는

입니다. 이 모델에서  $v_0$ 는 최소한 다음의 수익을 올립니다.

$$\frac{M \lfloor \log_2 N \rfloor}{N}; \quad (40)$$

$v_1$ 은 최소한  $\frac{M(\lfloor \log_2 N \rfloor - 1)}{N}$ 의 수익을 얻으며  $v_k$ 는 최소한 다음과 같은 수익을 올립니다.

$$\frac{M(\lfloor \log_2 N \rfloor - \lceil \log_2 (k+1) \rceil)}{N}. \quad (41)$$

$v_k$ 가 최소한  $\frac{M}{N}$ 의 수익을 올릴 수 있는 가장 큰  $k$ 는 다음과 같을 때입니다.

$$\lfloor \log_2 N \rfloor - \lceil \log_2 (k+1) \rceil \geq 1 \quad (42)$$

이것은 다음과 같은 의미를 가집니다.

$$\log_2 (k+1) \leq \log_2 \frac{N}{2}. \quad (43)$$

그러므로 최소한  $\frac{M}{N}$ 의 수익을 올릴 수 있는 k의 최대값은  $k = \lfloor \frac{N}{2} \rfloor - 1$ 입니다. 평균적으로 각각은의 수익을 올립니다.

$$\frac{M - \frac{M}{N}}{2^{\lceil \log_2 N \rceil - 1}} = \frac{M(1 - \frac{1}{N})}{2^{\lceil \log_2 N \rceil - 1}}. \quad (44)$$

분자는  $C$ 에서  $t = 0$ 이기 때문에  $v_0$ 의 엔터테인먼트 예산을 제외하는  $M - \frac{M}{N}$ 입니다. 분모는  $t = \lceil \log_2 N \rceil - 1$ ,  $|S| = 2^{\lceil \log_2 N \rceil - 1}$ 에 있기 때문에  $2^{\lceil \log_2 N \rceil - 1}$ 이며 이 점에서 S는 이 과정 전체를 통해 보상을 얻을 가능성이 있는 모든 노드로 구성됩니다. 이것은 보상을 얻을 수 없게 되는  $N - 2^{\lceil \log_2 N \rceil - 1}$ 개의 노드가 있다는 것을 의미합니다.

## 예제

미국인의 약 80%는 인터넷에 접속할 수 있는 컴퓨터를 보유하고 있습니다 [2]. 미국에는 3억 2,700만 명의 [3] 미국인이 살고 있으므로 인터넷에 연결된 기기가 있는 미국인은  $(0.8)(327M) = 261.6M$ 입니다. 한 명이 한 개의 기기를 보유하고 있다고 가정하여  $N = 261.6M$ 이라고 하겠습니다. 연간 미국 엔터테인먼트 소비량은 7,340억 달러입니다 [4] [5]. 미래의 이러한 지출의 대부분을 디지털로 가정해 보겠습니다. 하지만 인터넷에 접속할 수 있는 미국인의 80%만 예산에 포함하도록 하겠습니다. 이런 경우 지출액은  $(0.8)(734B) = \$587B$ 입니다. 지출의 10%는 유통 비용을 확보하기 위한 것으로 두겠습니다. 그러면  $M = (0.1)(\$587B) = \$58.7B$ 입니다. 그러면  $p = M/N = \$58.7B/261.6M = \$224.39$ 입니다. (44)에 의해 연간 평균 수익은 노드 당  $\$437.35$ 입니다. (40)에 의해  $v_0$ 가 별 수 있는 가장 큰 수익은  $\$6282.87$ 입니다. 그러므로 이 예제는 콘텐츠를 공유하는 초기의 피어가 가장 많은 보상을 얻는다는 것을 보여줍니다.

## 참고 문헌

- [1] Wikipedia (2018, June 19), *Complete graph*, [https://en.wikipedia.org/wiki/Complete\\_graph](https://en.wikipedia.org/wiki/Complete_graph)
- [2] C. Ryan and J. M. Lewis, “Computer and Internet Use in the United States: 2015,” *American Community Survey Reports* U.S. Census Bureau, September 2017 <https://www.census.gov/content/dam/Census/library/publications/2017/acs/acs-37.pdf>
- [3] World Population Review (2018, June 18) *United States Population 2018* <http://worldpopulationreview.com/countries/united-states-population/>
- [4] SelectUSA (2018, August 22), *MEDIA AND ENTERTAINMENT SPOTLIGHT*, <https://www.selectusa.gov/media-entertainment-industry-united-states>
- [5] Bureau of Labor Statistics (2017, August 29), *CONSUMER EXPENDITURES-2016* <https://www.bls.gov/news.release/cesan.nr0.htm>