ECO 퍼블릭 체인 성능 및 특성 분석

Akis Chalkidis akis@ecoc.io

October 2018

Contents

1 서 문

이 논문에서 우리는 ECO 체인의 특성을 분석 할 것입니다. 보다 구체적으로, 우리는 트랜잭션 속도, 확장성 문제, 블록 생성 시간의 평균 및 변화, 체인 공격, 고아 블록 및 네트워크 효율성, 공식 검증 등을 설명 할 것입니다.

2 거래 속도

많은 dApp을 호스팅할 계획인 체인의 관심은 체인의 용량이다; 즉, 초당 얼마나 많은 트랜잭션(TpS)을 수신할 수 있는가 이다. 여기서 우리는 최대 TpS와 지속 가능한 TpS, 즉 체인을 사용할 수 있게 유지할 수 있는 장기간의 평균 TpS라는 두 변수를 분리해야 한다.
먼저 최대 TpS를 분석해 보면, 실제 매개 변수와 실제 제한 사항에 대해 설명하고 거래 속도를 제한한다.

2.1 최대 TpS

Now, let:

h 홉(hop)의 수

b 초당 비트 수의 대역폭

s 최대 블록 크기 (바이트)

n 노드 수

c 각 노드에 대한 아웃 바운드 연결 수

ts 요구하는 트랜잭션 속도 (TpS)

l 바이트 단위의 평균 트랜잭션 크기

먼저, 모든 노드가 데이터를 얻을 때까지 필요한 홉(hop) 수를 계산한다. 연결 수가 c 이고 고정되어 있기 때문에 데이터(블록: block)가 있는 각 노드는 연결된 다른 노드로 브로드캐스트 되므로 플로우는 "스노볼" 효과를 따르게 된다. 수학에서 이것은 기하학적 진행이다:

$$a_h = a_0 c^{h-1}$$

여기서 a_h 는 $h-1$ 희망 후에 통보되는 노드의 수다. 여기서 a_0 은 블록을 형성(wins)하고 브로드캐스트를 시작할 준비가 된 노드이므로 $a_0 = 1$ 이라고 가정한다. h 홉 이후의 총 정보 노드 n 은

$$\sum_{a_h} a_0(1 - c_h) \; 1 \quad \sum a_h = \frac{1 - c^h}{1 - c} \tag{1}$$

$$1 - c \qquad \Rightarrow$$

그리고 모든 노드에 대한 정보가 전달될 희망 h 는 불평등으로부터 계산 될 수 있다.

$$\sum_{a_h \geq n} a_h \geq n \Rightarrow \frac{1 - c^h}{1 - c} \geq n \Leftrightarrow \frac{c^h - 1}{c - 1} \geq n$$

그리고 $c > 1 \Leftrightarrow c - 1 > 0$ 이기 때문에:

$$c^h \geq n\frac{c-1}{c-1} + 1 \Leftrightarrow \log_c h \geq \log_c (nc - n + 1) \Leftrightarrow h^{-1} \geq n \wedge c > 1 \Rightarrow c^c$$

$$\Leftrightarrow h \geq \log_c (nc - n + 1)$$

홉 수 h 는 정수이며 첫 번째 홉은 실제로 기하학적 진행의 $a_1 \rightarrow a_2$ 항에 대해 발생한다는 점을 명심해야 한다.

요컨대, $h = h + 1 - 1 \Leftrightarrow h = h \Rightarrow h = \log_c(nc - n + 1)$. 우리는 결국

$$h = \log_c (nc - n + 1) \quad (2)$$

을 얻는다.

우리는 지금 h 를 알고 있으므로 원하는 트랜잭션 속도 ts 를 달성하기 위해 가장 낮은 대역폭 요구 사항 b 를 계속 찾을 수 있다. 여기서 ts, 는 초당 최대 트랜잭션 수다. 여기서 우리는 최대 블록 크기 s 가 블록에 비해 매우 크다고 가정한다.

여기서 $bl/s \approx 0$ 헤더는 즉 bl 은 블록 헤더의 길이이다. 분명히, 블록 당 최대 트랜잭션 수는 $tr \approx s/l$ 이다. bt 를 대상 (평균) 블록 생성시간 (초)으로 정의하자.

다음을 얻는다:

$$tr\ ts \geq bt$$

하나의 "푸시(push)"에 필요한 전파 시간, 즉 하나의 홉(hop) 다음과 같다.:

$$T_i = T_l + T_p$$

여기서 T_i 는 각 홉을 완료하는 데 필요한 총 시간(초)이고, T_l 은 대기 시간이며, T_p 는 블록 데이터를 브로드캐스트하는 데 필요한 시간이다. 이 시간은 방송사의 업로드 대역폭과 수신기의 다운로드 대역폭에 따라 달라진다. 지연 시간은 주로 노드의 위치(네트워크의 물리적 토폴로지)에 따라 다르다. 큰 블록크기 데이터의 경우 T_p 가 T_l 보다 상당히 크며, 다시 말해 $T_i \approx T_p$ 라고 가정한다. 이것은 매우 정확하지는 않지만 분석 단순화에 도움이 될 것이다.
홉 h_i 의 최대 시간 $T_i \approx T_p = 8\frac{s}{b}$ 경우 바이트카 8 비트이고 s가 바이트 단위로 측정되므로 최대 시간이 된다. 총 전파 시간, t_T 는

$$t_T = \sum T_i \approx \sum T_p \frac{s}{b} \sum 8 = 8 \sum \frac{s}{b}$$

여기서는 모든 조건이 동일하기 때문에 (평균 대역폭, 전파될 블록 크기) 각 홉 h_i 에 동일한 전파 시간이 필요하다는 가정을 한 번 더 할 수 있다. 결과적으로,

$$t \approx 8 \sum_{T} \frac{s}{b} = 8h \frac{s}{b} \quad (i=1 \text{ 부터 } h)$$

우리는 이미 $h = \log_C (nc - n + 1)$ 임을 증명 했으므로,

$$t_T \approx 8h \frac{s}{b} \Rightarrow t_T \approx \frac{8s \log_C (nc - n + 1)}{b} \quad (3)$$

위의 근사값을 계산하는 것은 전혀 어렵지 않지만 유익하다. 총 시간 t_T 가 최대 블록 크기 s , 연결 수 c , 노드 수 n 및 평균 네트워크 속도 (대역폭) b 에 달려 있음을 분명히 보여준다. 따라서 블록 생성 시간 bt 에 의존하지 않는다. 그러나 bt 는 분명한 한계를 설정한다: *gossip protocol*

$$t_T < bt \quad (4)$$

왜 그런지 쉽게 알 수 있다. 노드는 블록 데이터를 "다운로드" 할 수 있는 충분한 시간을 가져야한다. 따라서, 새로운 노드가 생성되는 동안 일부 노드가 이전 블록에서 데이터를 가져오는 것은 허용되지만 오래 지속되는 것은 안전하지 않다. 총 전파 시간은 블록의 평균 생성 시간보다 낮아야 한다. (3)과 (4)의 결합으로,

$$t_T \approx \frac{8s\log_c(nc - n + 1)}{b} \Rightarrow bt > \frac{8s\log_c(nc - n + 1)}{b} \Leftrightarrow$$

$$\Leftrightarrow b > \frac{8s\log_c(nc - n + 1)}{bt} \Leftrightarrow$$

$$\Leftrightarrow b_{min} = \frac{8s\log_c(nc - n + 1)}{bt} \quad (5)$$

균등한 (5)는 네트워크가 크기 s의 블록 크기를 바이트 단위로 유지해야 한다는 점에서 가장 낮은 대역폭 b_{min}을 분명히 보여준다. 네트워크 n의 노드 연결 수와 연결 c의 사이에 로그 관계가 있기 때문에 데이터의 프로포게이션이 쉽게 확장된다는 점도 주목할 필요가 있다.

즉, 홉 $h(n)$ 의 수는 (log n)이다. 이러한 효율성의 이유는 포로포게이션(propogation)이 가십 프로토콜 *(gossip protocol)*에 기초하기 때문이다 [1]. 비트코인과 에코체인은 (기본) 연결 수 c = 8로 가십 프로토콜을 따른다. 그것은 **src/net.h** 파일의 코드에서 쉽게 볼 수 있다:

```
static const int MAX_OUTBOUND_CONNECTIONS = 8;
```

위에서 우리는 c = 8이라고 결론을 내렸다. 참고로, 노드가 아웃 바운드 연결 수를 자유롭게 변경할 수 있다는 덤을 강조해야 한다. 즉, c는 합의 프로토콜의 매개 변수가 아니다. 위의 코드 라인을 8개에서 임의의 숫자로 바꾸는 것만은 네트워크가 받아들일 수 있는데, 실제로 감지할 수 없기 때문이다.

예를 들어, 노드는 높은 럽도드 대역폭을 가지고 있을 수 있고 또한 네트워크를 돕고 싶을

수 있으므로 c를 바꾸는 것은 쉬운 소프트 포크(soft fork)다. 우리는 여기서 대부분의 노드가 c를 변경할 동기가 없기 때문에 c를 변경하지 않을 것이라고 안전하게 가정할 수 있다. c의 기본값이 8보다 큰 값으로 설정되어 있는 경우, 반드시 더 빠른 전파 시간을 가져오지는 않는다.

데이터베이스 커밋 시 디스크 쓰기 시간 및 트랜잭션 수가 많은 경우 필요한 유효성 검사 시간 때문에 CPU 지연 시간이 제한된다. 그래서 ECOChain은 이미 언급했듯이 노드를 제한하지 않는 $c = 8$을 유지한다; 각 노드는 매우 쉽게 변경될 수 있다.

숫자로 예를 들어 보자: 최대 블록 크기 $s = 4Mbytes$, 블록 시간 $bt = 32sec$, 노드수 $n = 4,000$ 연결 수 $c = 8$ 의 경우 등식 (5)의 최소 네트워크 속도(대역폭)는 다음과 같아야 한다.

$$b_{min} = \frac{8s\log_c(nc - n + 1)}{bt} = \frac{8 * 4000000 * \log_8(4000 * 8 - 4000 + 1)}{32}\frac{bits}{sec} =$$

$$= 1000000 * \log_8 28001\frac{bits}{sec} = 1000000 * 4.92439691020751\frac{bits}{sec} =$$

$$= 1000000 * 4\frac{bits}{sec} = 4000000 = 4Mbps$$

As we have already seen the function $b_{min}(n)$ is logarithmic. So with the same parametres but with a much larger number of nodes $n = 30,000$ for example we have

$$b_{min} = \frac{8s\log_c(nc - n + 1)}{bt} = \frac{8 * 4000000 * \log_8(30000 * 8 - 30000 + 1)}{32}\frac{bits}{sec} =$$

$$= 1000000 * \log_8 210001\frac{bits}{sec} = 1000000 * 5.8933455574294\frac{bits}{sec} =$$

$$= 1000000 * 5\frac{bits}{sec} = 5000000 = 5Mbps$$

오늘날 (Q4 2018) 글로벌 평균 대역폭이 10Mbps 안팎이다. 또한, 우리는 콩신의 기술이 발전함에 따라 대역폭은 더 높아질 것이라는 것을 명심해야 한다. 대역폭의 증가는 닐슨 법칙*Nielsen's Law*[2]에 따른다는 믿음이 있는데, 이 법칙은 글로벌 대역폭이 매년 50% 정도 증가한다고 명시하고 있다. 이 전망은 다소 낙관적인데, 과거 자료가 연간 금리 인상 속도를 더디게 보여주고 있기 때문이다. 그런데 사실은 인상률이 상당하다. 따라서 일부 사람들이 오늘날 4Mbps 또는 5Mbps 를 근소한 차이로 달성할 수 있다는 것은 분명히 해야 한다.

마지막으로 위의 파라미터에 대한 *TpS*를 계산해 보자. 블록 크기 $s = 4MB$의 경우, 블록 시간 $bt = 32seconds$ 이고 $l \approx 200$이라고 가정하면 최대 트랜잭션 속도는 다음과 같다.

$$ts \geq \frac{tr}{bt} \approx \frac{\frac{s}{1}}{bt} = \frac{s}{l * bt} \approx \frac{4 * 10^6}{200 * 32} = \frac{10^6}{200 * 8} = \frac{10^6}{1600} \approx 625$$

따라서 최대 TpS $ts \approx 625$ 트랜잭션/초. 이것은 이론상 속도다. 실제로 초당 560 건 정도의 거래량을 측정했다. 테스트는 AWS cloud에서 적은 수의 노드($n = 100$) 에서 실행되었다. 실제 네트워크 프로토콜은 긴 시간 초과를 피할 수 있는 기능이 내장되어 있다. 노드가 응답을 받지 못하거나 2 초 이상 매우 느린 연결을 수신하는 경우, 노드는 연결을 끊고 다른 노드에 연결하며, 그렇게 하면 느린 전파 시간을 피할 수 있다.

독자는 위의 모든 내용이 근사치임을 명심해야 한다. 분석을 단순화하기 위해 특정 가정 하에서 이러한 결과를 결론지었다. 예를 들어, 블록 헤더 크기는 본문 크기와 비교하여 무시할 수 있다고 가정했다 (사실). 또한, 지연 시간 (T1)은 데이터 (T p)를 전송하는 시간에 비해 매우 짧다고 가정했다. 이것은 더 많은 주의를 기울일 가치가 있다. 사용중인 프로토콜은 잘 알려진 "3방향 핸드쉐이크(three-way handshake)"(SYN, SYN-ACK, ACK)를 요구하는 TCP이다. 여기서 노드 연결은 대륙 간 연결이기 때문에 지연시간은 보통 80-120ms (밀리 초)사이라고 가정한다. 일부 연결은 대륙간 일 것이다. 이러한 몇 가지 트랜잭션의 경우 대기 시간은 약 200ms가 된다. 네트워크에 기술적 문제 (매우 자주 노드 연결 및 연결 끊김)가 있는 경우 대기 시간이 길어 지므로 최종 TpS 결과에서 약간의 역할을 수행 할 수 있다. 노드 수가 많은 네트워크의 경우 대규모 대기 시간 (많은 노드를 감염시키는 높은 대기 시간)을 검증할 수 없다. 실제로 대부분의 노드에서 연결 문제가 발생하는 것은 불가능하다. 세 번째 가정은 평균 트랜잭션 규모입니다. 가장 일반적인 트랜잭션에는 vin과 두 개의 vout이 있다 (하나의 수신기와 소유자에게 반환되는 "변경"). 이것은 보통 191바이트 길이다. 그러나 더 많은 vout을 가진 트랜잭션이 있을 것이다. 스마트 컨트랙트(예: 긴 바이트코드 스마트 컨트랙트가 배치 된 경우) 의 경우도 있다. 평균 트랜잭션 규모를 예측하기는 정말 어렵다. 측정 가능한 것은 트랜잭션이 아니라 처리량이다. 200bytes의 평균 트랜잭션 크기의 추정이 낙관적이라면 TpS는 다소 낮을 수 있다.
우리가 왜 높은 블록 크기 제한을 선택하는지에 대한 마지막 참고 사항. 블록의 최대 크기는 문제없이 피어(peer)의 코드에서 더 낮게 설정할 수 있으며, 실제로 다른 피어(peer)는 각 노드에서 내부적으로 설정된 크기 제한을 알 수 없다. 예를 들어, 소프트 포크는 src / ecoc / ecoc.h 파일의 코드 내에서 라인을 바꾸는 것만으로 블록 크기를 줄일 수 있다:

```
const int blockSizeLimit = 4000000 ; // blocksize limit
```

그 반대는 사실이 아니다. 노드가 블록 크기 제한을 늘리려면 실제로 할 수 없다. 단지 크기를 늘리고, 마이닝하고, 일반적인 허용 한계를 넘어 블록을 형성하는 것만으로, 합의 프로토콜 규칙에 위배되므로 다른 노드로부터 거부로 이어질 수 있다. 다시 말해, 블록 크기 제한을 늘리려면 하드 포크가 필요하며, 모든 사용자는 클라언트 버전을 업데이트해야 변경사항이 적용될 수 있다.
이것은 우리가 처음에 높은 블록 크기 제한을 선호하는 또 다른 이유다.

2.2 지속 가능한 TpS

경제적 의미에서 확장성 문제에 대해 이야기 할 때다. 이전 섹션에서는 블록체인 용량에 대한 기술적 제한을 분석했다. 이 섹션에서는 실제 경제가 설정하는 제한을 분석 할 것이다. 전체 노드를 실행하는 실제 비용은 블록체인의 전체 크기에 따라 선형적으로 증가한다. 퍼블릭 체인의 데이터는 빠른 속도로 증가한다. 이 속도는 초당 평균 트랜잭션 수에 따라 다르며 초당 바이트 수로 측정된다. 시간 단위당 크기가 높은 경우, 체인(node:노드)의 보존자에 다음과 같은 새가지 문제가 발생한다:

(a) 초기 다운로드 시간이 너무 오래 걸릴 수 있음
(b) 많은 RAM이 필요함
(c) 많은 저장 공간(디스크 공간)이 필요함

이러한 각 문제를 자세히 살펴 보자. (a)의 경우, 문제는 노드가 공개 체인에 처음 가입하려고 할 때 모든 데이터 (이력)를 다운로드해야 한다는 것이다. 크기가 매우 크고 노드의 다운로드 속도가 느린 경우 초기 다운로드가 완료 될 때까지 오랜 시간(수일)이 걸릴 수 있다.
이로 인해 새로운 노드가 퍼블릭 체인에 가입하지 못하게 될 수 있다.
(b)의 경우, UTXO 모델은 트랜잭션 수에 선형인 RAM이 필요하기 때문에 대규모 체인은 노드에서 높은 RAM (메모리) 사용 기능을 요구한다. RAM은 디스크 저장소보다 훨씬 비싸다. 따라서 메모리 스왑 옵션을 고려할 때 노드가 가상 메모리 (디스크 공간을 RAM)로 사용하는 것이 당연해 보인다. 오늘날 데스크탑용 모든 운영 체제 (Windows, Linux, Mac 및 모든 * nix 시스템)는 이 기능을 제공한다. 그러나 거기에는 함정이 있다: 디스크가 메모리보다 훨씬 느리다. 이 사실은 다음과 같은 문제를 가져온다: 스테이커(staker)는 블록의 유효성을 검사하는 데 더 많은 시간이 필요하므로 시작하기 전에 몇 초 지연시켜야 한다.
채굴(스테이킹). 이것은 PoW 프로토콜에서 문제가 되지만, 세분성이 검증 시간보다 길면 PoS에서 실제 문제를 일으키지 않는다. 즉, 스테이커를 불리하게 만들지 않는다. 가상 메모리가 사용된다는 점을 감안할 때, 이 문제, (6)은 우리가 제시할 (c)와 동일하다고 가정할 수 있다.
(c) 체인 크기의 성장률은 실제 (평균) 거래 속도에 따라 달라진다. 예를 들어, 평균 트랜잭션 속도가 초당 50 트랜잭션이고 트랜잭션 당 평균 길이가 200 바이트라고 가정하면 1 년 동안 크기 데이터는 더 커진다(몸에 비해 매우 작기 때문에 방정식의 블록 헤더를 계산하지 않는다).

$$\Delta S = tps * l * t = 50 * 200 * 60 * 60 * 24 * 365 = 315360000000 bytes \approx 315 ttb$$

상당한 양의 데이터다. 따라서 분산화를 유지하면서 네트워크에 노드를 유지할 수 있는 지속 가능한 TpS를 찾아야 한다. 사이드 노드로서, 전에 언급 된 적이 없기 때문에, 우리는 스테이커(staker)가 민트(mint)를 만들 기회를 얻고 이득을 얻으려면 전체 노드를 운영해야 한다는 점을 강조해야 한다. 그래서 스테이커가 자신의 비용(합리적 경제 활동)보다 더 큰 기대 이득이 되는 한 노드를 운영하는 네크워크에 참여할 동기를 부여한다. 비용은 하드웨어(기본적으로 디스크 공간)와 통신 비용이다.

여기서 결정적인 요소는 디스크 공간 비용이다. 이미 설명했듯이 RAM 제한 리소스는 디스크 공간 리소스로 변환 될 수 있다. 코인 가격에 따라 미래의 비용이나 스테이커의 이윤을 정확히 예측할 수는 없지만 기술이 발전함에 따라 디스크 저장공간 비용의 감소와 관련하여 체인의 크기 성장률 사이의 연결을 찾을 수 있다. 즉, 몇 가지 가정 하에서 비용 함수 c (c0, y)를 찾을 수 있는데, 그 값은 스테이크의 최대 허용 비용보다 낮아야 한다, cmax라고 하자. 여기서 c0은 스테이커가 네트워크에 가입하는 시점의 비용이고 y는 비용이 c0인 시점 이후의 시간이다. 따라서 우리의 첫 번째 제한 사항은 다음과 같다:

$$c_{max} > c(c_0, y) \quad (6)$$

Now, let:
$c_{max,I}$ 는 staker I에 대해 허용되는 최대 비용이다 (let's say in USD)
c_0 은 처음 가입할 때의 노드 실행 비용 (USD)
s_0 시작 연도에 실행할 노드의 최소 디스크 크기(GBytes 단위)
dc 디스크 저장공간 비용 (USD/Gbyte 단위)
s_y 연도 y(가입 후 y년) 에 실행할 노드의 최소 디스크 크기(GBytes 단위)
gr 체인 증가율 (GBytes/year로 측정됨)

$gr = \overline{tps} * l$, 여기서 tps 는 평균 트랜잭션 속도를 의미하고 l은 평균 트랙잭션 길이를 나타낸다. 여기서 b_h 는 블록헤더 크기이고 b_b 는 본문 크기이다. 분명히, 정의에 $c(c_0, 0) = c_0$ 이다. $b_b \approx 0$ 노드에 대한 1년 후의 새로운 비용은

$$c_1 = c_0 - \delta c_{y0 \rightarrow y1} + \delta s_{y0 \rightarrow y1} * dc_1$$

c_1 is the cost to run a node after on year of first join (in USD) $\delta c_{y0 \rightarrow y1}$은 기술 발전으로 인한 비용 감소, 즉 시간이 지남에 따라 비용 절감 효과가 있다.
$\delta s_{y0 \rightarrow y1}$ 은 1 년 후의 추가 체인 데이터 크기이다
dc_1 y_1의 디스크 저장공간 비용 (in USD/Gbyte)
새로운 최대 허용 비용은 $c_{max,1}$ 이며 $c_{max,0}$ 보다 높거나 낮을 수 있다. 코인가격이 낮아지거나 스테이커의 잔고가 적어서 예상되는 보상이 낮아질 수 있기 때문이다. Y 년의 연간 비용 차이와 최대 허용 비용 차이를 $\delta c_{y,y-1}$ 및 $\delta c_{max\,y,y-1}$ 로 지각적으로 상징 할 것이다.
가장 먼저 주목해야 할 것은 균형이 매우 낮은 (0에 가까운) 스테이커들은 완전한 노드를 운영할 경제적 동기가 없다는 것이다. 즉, 언제든지 y는 총 현재 잔액 bi, 스테이킹 보상 r, 그리고 코인과 USD 사이의 환율 e를 가진 스테이커가 노드를 실행하기 위해 다음과 같은 불평등을 나타낸다.

$$\delta c_{y,y-1} < \delta pr_{y,y-1} \quad (7)$$

여기서 $pr = b_i \; r \; e$ 은 채굴자의 지분 수익입니다. 일반적인 영어로, 연간 수익은 연간 추가 비용을 충당해야 한다. 수익이 환율 *e에 의존한다는 것이 명백하지만, e가 r과 독립적이기 않기 때문에 스테이킹 보상 r에 의존한다고 결론을 내릴 수 없다. r*이 환율을 낮추는 인플레이션을 생성하기 때문에 $e(r)$ 는 단조 감소 함수다. 다시 말해, *r 값이 높을수록 지분을 가진자에게 더 많은 수익을 보장하지는 않는다. 부수적으로,* $\delta c_{y,y-1}$ 는 음수일 수 있다. 얼핏 보기에는 이상하게 보일 수 있다; 체인 데이터가 증가할 때 노드 비용이 어떻게 감소할 수 있는가? 그러나 기술 혁신이 이루어지면 저장공간 비용 dc_y *는 크게 떨어질 것이다. 이 경우,*

$$\delta c_{y-1 \to y} > \delta s_{y-1 \to y} * dc_y$$

그럼 (then)

$$c_y = c_{y-1} - \delta c_{y-1 \to y} + \delta s_{y-1 \to y} * dc_y \wedge \delta c_{y-1 \to y} > \delta s_{y-1 \to y} * dc_y \Rightarrow$$

$$\Rightarrow c_y < c_{y-1} \Leftrightarrow c_y - c_{y-1} < 0 \Leftrightarrow \delta c_{y-1 \to y} < 0$$

따라서 체인 크기가 증가하는 동안 패스 오프 시간에 따라 총 비용이 감소 할 수도 있다는 것은 분명하다. 이 경우, **미미한** 연간 수익이 감소하지만 **미미한** 연간 비용 절감보다 적더라도, 스테이커는 상주하며 전체 노드를 운영하는 것이 여전히 수익성이 있다. 요컨대, 중요한 요소는 한계비용과 채굴로 인한 한계 이익이다. 여기서 한계 값은 연간 가치 차이를 의미한다.
좀 더 정확하게 하자. 채굴자는 자신의 미래 보상이나 미래 비용을 모른다. 우리는 **예상** 연간 수익과 **예상** 연간 비용에 대해 이야기하고 있다. 경제 이론에서 우리는 합리적인 경제적 행동을 위해 한계 비용이 (장비에 대한 최초의 초기 투자 후) 한계 이익과 같거나 적어야 한다는 것을 알고 있다. 따라서 다음과 같은 불평등은 스테이크 홀더에게 있다:

$$\overline{Mc_y} < \overline{Mpr_y} \quad (8)$$

스테이커 i $\mathrm{Mpr}_{y,i}$ 에 대한 한계 연간 이윤은 이미 살펴본 바와 같이, 그의 균형 b_y**i** , 현재 교환율 e_y 및 연간 스테이킹 보상 *r*에 달려 있다. 반면, 연간 한계 비용은 퍼블릭 체인 *gr*의 성장률과 디스크 저장공간 dc_y 단위당 새로운 비용에 따라 잘라진다. 따라서 위의 불평등은

$$\overline{Mc(gr,\ dc_y)} < \overline{Mpr(b_y\mathbf{i}\ ,\ r,\ e_i)} \quad (9)\ \overline{Mc(gr,\ dc_y)}$$

에 대한 분석 능력 향상:

$$\overline{Mc(gr,\ dc_y)} = c_y - c_{y-1} \wedge c_y = c_{y-1} - \delta dc * s_{y-1} + \delta s_y * dc_y \Rightarrow$$

$$\Rightarrow \overline{Mc(gr,\ dc_y) = \delta s_y} * dc_y - \delta dc * s_{y-1} \Leftrightarrow$$

$$\overline{Mc(gr,\ dc_y) = gr} * dc_y - \delta dc * s_{y-1} \Leftrightarrow$$

gr 은 체인의 성장률, dc_y 는 y 년의 저장공간 비용, δ 는 단위 저장공간당 기술 비용 감소 및 s_{y-1} 은 전년도의 저장공간 크기다. 결국 우리는:

$$Mpr(b_y\boldsymbol{i}, r, e_i) > gr * dc_y - \delta dc * s_{y-1} \quad (10)$$

모든 합리적인 스테이커(staker)가 스테이크를 계속 유지하기 위해, 즉 전체 노드를 계속 실행하려면 위의 불평등이 유지되어야 한다. 따라서 그의 결정은 그의 지분(계정 잔고), 스테이킹에 대한 환율 및 연간 보상, 체인의 성장률 및 디스크 저장공간의 비용 절감 요소(기술 발전에 따라 다름)를 기반으로 한다. 불행히도 더 단순화 할 순 없다. 왜냐하면 이것이 부정확한 결과를 초래할 수 있기 때문이다.

예를 들어보자. 체인의 성장이 지난해 $gr = 315 ttb$ (이전의 예와 같이)이고 현재 모멘트(연간)의 1Gb 비용은 $dc_* 1 ttB = \$0.04$, 이고 전년도 비용은 $dc_* 1 ttB = \$0.05$ 이며 전년도 체인의 총 크기는, $s_{y-1} = 1.5 Tb$ 라고 가정해 보자. 그래서 지난 1년간 그의 한계비용은

$$Mc(gr, dc_y) = gr * dc_y - \delta dc * s_{y-1} \Rightarrow$$
$$\Rightarrow Mc(gr, dc_y) = 315 ttb * 0.04\$/ttb - (0.05\$ - 0.04\$)\$/ttb * 1.5 * 1000 \Leftrightarrow$$
$$\Leftrightarrow Mc(gr, dc_y) = -2.4\$ USD$$

이 예에서는 추가 연간 비용이 마이너스이므로 작년 비용에서 2.4$ 만큼 낮아 지므로 작년의 추가 이익은 2.4$를 넘지 않을 수 없다:

$$Mpr(b_y\boldsymbol{i}, r, e_i) > -2.4\$ \Rightarrow pr_y - pr_{y-1} > -2.4\$ \Leftrightarrow pr_y + 2.4\$ > pr_{y-1}$$

저장공간에서 예상치 못한 기술 혁신이 발생하거나 체인 데이터의 성장률이 너무 낮은 경우를 제외하고는 일반적으로 한계 비용이 긍정적일 것이다. 어쨌든 추가 이익은 추가 비용을 충당해야 한다.

위의 예에서 스테이커가 처음으로 네트워크에 참여하는 최대 비용을 확인해보자. $c_{max,0} = s_0 * dc_0$ $\Rightarrow c_{max,0} = 1.5 Tb * 0.05 USD/ttb\ Gb \Leftrightarrow c_{max,0} = 1.5 * 1000 * 0.05 USD \Leftrightarrow c_{max,0} = 75 USD$ $pr_{min,0} = 75 USD$. 따라서 그의 예상 수익은 노드 운영을 결정하기 위해 $75 USD$ *이상이 되어야 한다*.

지금까지의 분석에서는 네트워킹 비용을 제외했다. 이 단순화는 우리의 일을 단순하지만 더 부정확하게 만든다. 진실은 많은 컴퓨터에서 사용자가 어느 방식으로든 네트워크 트래픽에 대해 비용을 지불한다는 것이다. 이 경우 그의 네트워크 비용은 0이다. 일반적인 예로는 이미 개인 용도로 인터넷을 사용하는 데스크탑 사용자나 다양한 다른 일을 하고 있고 사용되지 않는 대역폭이 많은 서버가 있다. 현실 세계에는 "불합리한 행동"도 있다. 스테이커(Staker)는 자신의 계정과 균형이 충분하지만 약간의 손실이 발생하더라도 노드 실행을 결정하기 위해 노드 또는 그 반대로 실행하지 않기로 결정할 수 있다. 또한, 우리의 방정식과 불평등은 예측할 수 없는 미래 환율에 따라 달려있다. 이 섹션에서는 합리적 결정에 중요한 역할을 하는 요인과 체인에 합류할 시기 또는 노드 실행을 중단할 동기가 있을 때의 관계를 수학적으로 보여 주었다.

우리가 명심해야 할 것은 분산화는 노드의 수에 달려 있으며, 이는 결국 코인의 가치(높을수록, 노드 가입이 많을수록)와 저장공간 비용 감소율(다시 말해, 비용 감소율이 높을수록 노드 가입이 많을수록)에 따라 달라집니다. "크라이더의 법칙*(Kryder's Law)* "[?]이 있는데, 이것은 실제로 법이 아니라 2020년의 디스크 저장 비용에 대한 예측이다. 예측은 매년 약 40%의 감소율을 보이고 있다. 지금까지 그의 예측은 지나치게 낙관적이지만, 예측할 수 없는 속도로 저장공간 비용이 실제로 감소하고 있다는 것이다. 예를 들어 양자 저장 또는 DNA 사용 분야에서 혁신이 크게 일어나는 등 미래의 어딘가에서 원가의 급격한 감소가 일어날 수 있다는 것과 다르지 않다. 인수는 체인의 성장률이 자신의 한계 이윤 이상으로 스테이크 비용을 증가시킬 수 있다는 것이다. 이 섹션에서는 저장공간 비용 감소에 크게 달라진 다는 것을 증명했다.

3 블록 생성 시간

PoS 아키텍처가 어떻게 합의에 도달하는지 간단히 설명하겠다. PoW 프로세스의 모방이다. 차이점은 PoS에서 "채굴자(miner)"는 PoW의 경우와 마찬가지로 해시함수에 어떤 값(인수)도 전달할 수 없다는 것이다. 시간(타임 스탬프)과 공개 주소(양수 계정 잔액이 있어야 함)만 통과할 수 있다. 물론, 다음 블록 트랜잭션이 형성될 때까지 공개 주소도 고정된다. 그의 유일한 옵션은 타임 스탬프를 변경하는 것이다. 타임 스탬프가 변경되면 해시가 생성된다. 블록을 이기기 위해 필요한 조건은 다음과 같다:

$$h(c1, c2, ..., cn, ts, pa) < t * b$$

여기서(Where):

h는 해시 함수의 결과
c1, c2, ..., cn은 변경할 수 없는 인수
ts는 타임 스탬프 (스테이커가 변경할 수 있는 유일한 것)
pa는 그의 퍼블릭 주소
t는 난이도(PoW와 같은)에 의해 설정되고, 고정된 수의 블록마다 재설정된 값(목표)
b는 그의 퍼블릭 주소에서의 균형

위의 불평등이 사실일 때, 스테이크 홀더는 다음 블록을 주장하고 형성할 수 있다. 각 개별 스테이브홀더가 각기 다른 타임 스탬프 값에서 블록을 이길 확률은 p i = p * b I 이다. 여기서 pi는 개인이 자신의 계정에 있는 잔액에 비례하여 블록을 획득할 확률이다. 각 타임 스탬프(초)마다 총 네트워크가 블록을 형성할 확률은 물론 각 "틱" 이라고 하자.

$$P = p_1 + p_2 + p_3 + \ldots + p_s = \sum_{1}^{s} p_i = \sum_{1}^{s} p * b_i = p \sum_{1}^{s} b_i = p * B$$

여기서 s 첨자는 개별 스테이크 홀더(그의 공개 주소로 식별)이고, B는 합의에 참여하는 모든 스테이크 홀더의 총 잔액, 즉 다음 블록을 구성하려고("win") 시도하는 것이다.

지금까지 우리는 확률 p가 얼마인지에 대해 논의하지 않았으므로 체인이 다음 "틱"에서 블록을 형성해야하는 최종 확률 를 계산할 수 있었다. 또한, 다음 순간이 아니라 다음 진드기에 대해 이야기하고 있다. 사실, 우리는 각 스테이크 홀더들에게 매 초가 아니라 $g+1$ 초마다 시도하도록 강요 할 수 있다. 이제부터 g를 세분화이라고 부를 것이다; $g+1$ 은 물론 정수이며, 그뿐만 아니라 $g+1 = 2^k \quad g = 2^k \quad 1$ 의 형식을 가지며, 여기서 *k도 정수다.* $k = 0$ 의 최대 세밀도를 가지고 있으며, 즉, $g = 0$, 이므로 스테이크가 매초마다 해시를 제출할 수 있다는 뜻이다.

우선, 세분화 작동 방식에 대해 자세히 설명하겠다. 그리고 나서, 세분화로 인해 체인에 보안이 추가되는 이유와 트레이드 오프가 무엇인지 설명 할 것이다. 스테이커들이 매 초 마다가 아니라 g + 초마다 해시를 계산하도록 하려면 타임 스탬프를 마스킹해야 한다. NAND 논리 연산을 타임 스탬프와 함께 사용하면 쉽게 수행 할 수 있다. 예를 들어, 4개의 순차적 타임 스탬프가 동일한 해시를 제공하도록 하려면 (입력이 동일 할 때 동일한 해시를 제공하므로 3개의 타임 스탬프를 쓸모 없게 렌더링) 마지막 두 비트의 AND 연산을 수행하면 된다. 0의 타임 스탬프 (즉, 이진 xb00). 따라서 이진11 (즉, 3) 로 NAND를 수행하면 타임 스탬프 (마지막 두 비트를 0으로 설정)를 마스킹하여 4개의 타임 스탬프를 모두 동일한 값으로 변환한다. 아래에서는 코드 (C++) 에서 이것이 어떻게 구현되는지 확인할 수 있다:

```
nTimeBlock &= ~STAKE_TIMESTAMP_MASK;
```

세분화가 $2^n \quad 1$ 형식인 이유는 분명하다. 타임 스탬프의 마지막 n 비트를 가리기 위해서다. 낮은 입도(높은 g 값)는 스테이크–그라인딩(stake-grinding) 공격을 방지하는 데 도움이 된다. 공격자가 성공적인 공격을 수행하는 것이 훨씬 어렵다. 낮은 입도의 의미는 블록 생성 시간에 대한 더 높은 변형이다. 우리는 평균 생성 시간과 블록 시간의 변화를 조사 할 것이다.

모델이 어느 확률 분포에 적합한 지 쉽게 알 수 있다. 스테이커는 g +1 초마다 블록을 형성할 수 있다. 여기서 g ∈ {0,1,3,7,15,31, ..., 2n-1} .g + 1은 2의 거듭 제곱이다. 따라서 모든 g + 1 초 그들은 블록을 형성하려고 한다. 스테이커의 평균 시간 및 누적 분포(CDF)가 블록을 형성하는 데 흥미가 있으므로 블록이 형성될 p 확률을 계산해야 한다. 이러한 시도는 우리의 순환 모델을 불연속적으로 만들고, 비트코인이나 다른 PoW 합의에서처럼 연속적이지 않게 만든다. 선택할 수 있는 정확한 확률 분포는 기하 분포[5] 또는 GD[5]이다. 각 시도마다 총 스테이킹 잔고가 거의 같고 평균 고정 시간 (목표시간)을 재조정하기 위해 난이도가 바뀐다는 가정하에, 우리는 이미 평균값 (평균 = 목표 시간)을 알고 있다고 가정한다.

기하 분포는 i 번의 시도 후 첫 번째 성공을 얻는 데 필요한 X 베르누이(Bernoulli) 시행의 분포를 계산하는 이산 확률 분포이다. 우리는 이미 알려진 g와 목표를 염두하고 있으며, 어떤 분포 모델이 우리의 모델을 정확하게 묘사하는지 명확하게 밝히면 확률 p, 시도 후 블록을 형성 할 누적 확률 등을 계산할 수 있다.
수학에서 **GD**의 평균이 다음과 같다는 것을 알고 있다:

$$E[p] = \frac{1}{p}$$

So we have: $E[p] = 1 \Rightarrow \frac{target}{} = \frac{1}{p} \Leftrightarrow p = \frac{g+1}{target}$

성공 시도의 첫 번째 발생, 즉 i 가 시도한 후 블록 생성은

$$P(X = i) = (1 - p)^{i-1}p$$

예를 들어, g = 7 , $target$ = 128 sec 및 i = 1 (첫 번째 시도)의 확률은 다음과 같다:

$$P(X = i) = (1 - p)^{i}p \Rightarrow P(X = 1) = (1 - 0.0625)^{1-1} * 0.0625 \Leftrightarrow$$
$$\Leftrightarrow P(X = 1) = 1 * 0.0625 \Leftrightarrow P(X = 1) = 6.25\%$$

자, 이제 $k = g+1$ 우리는 $i \leq k$ 인 경우에 흥미를 갖도록 하자. 왜냐하면 이것은 평균 블록 시간이 서 있거나 블록을 대부분 생성해야 하기 때문이다. 그보다 훨씬 길면 분명히 바람직하지 않다. 목표 시간까지 블록이 형성 될 확률은 GD에 대한 누적 분포 함수 (CDF)로부터 쉽게 계산 될 수 있다.

$$F_X(x) = P(X \leq x) \Rightarrow F_X(k) = 1-(1-p)^k \Rightarrow F_X(k) = 1 - \left(1 - \frac{g+1}{target}\right)^{g+1} \Leftrightarrow$$

$$\Leftrightarrow F_X(k) = 1 - \left(1 - \frac{g+1}{target}\right)^{g+1}$$

For the above example (g = 7 , $target$ = 128) $k = \frac{target}{g+1} = \frac{128}{7+1} = 16$ and

$$F_X(16) = 1 - \left(1 - \frac{128}{128}\right)^{7+1} = 1 - \left(\frac{120}{128}\right)^{16} \approx 64.39\%$$

따라서 평균 시간 또는 이전에 65.6% 의 확률로 블록이 형성 될 것으로 예상한다.

효과와 보안을 연구하기 위한 다음 섹션으로 넘어가기 전에 이는 모든 것이 k(네트워크에 대기 시간이 없다고 가정하는 경우)에 의존한다는 것을 보여준다. 즉 $k = \frac{target}{g+1}$ 각 스테이크 홀더가 목표 시간까지 시도한 횟수가 가장 결정적인 요인이다. k 값이 낮은 경우, CDF가 약간 더 높으며, 이는 바람직하다. 또한 스테이커를 제한하여 그라인드(grind) 타입의 공격을 어렵게 만들기 때문에 더 안전하다. 불행하게도 네트워크 효율성과 보안에 부정적인 영향을 미치는 더 많은 고아 블록(아래 참조)을 생산하기도 한다.

마지막으로 대기 시간이나 전파 시간이 없다고 가정한 것을 잊지 말아야 한다. 실제로는 대기 시간이 상당할 수 있으며 전파 시간(블록이 "푸시"될 시간)도 있다. 즉, 데이터 전송은 즉각적이지 않는다는 것을 의미한다. 연구[4]는 전 세계에 노드가 많은 네트워크인 비트코인의 경우, 평균 전파 시간은 12.6초 라는 것을 보여주었다. 일부 노드가 반응하는데 보통 시간보다 더 오래 걸리거나 심지어 연결이 끊어지기 때문에 자연수러운 현상이다. 섹션 2.1에서는 전파 시간에 대한 자체 분석 및 테스트를 수행했다. 독자는 이 섹션을 연구하여 ECOChain의 네트워크 동작에 대해 더 잘 이해해야 한다.

target	g	$k = {}_{target}$	p	p for $k = 1$	*CDF* for k	p_{or}%
32 sec	7	4	0.25	0.25	0.6836	6.25
64 sec	7	8	0.125	0.125	0.6564	1.56
64 sec	31	2	0.5	0.5	0.75	25
128 sec	1	64	0.015625	0.015625	0.635	0.02
128 sec	3	32	0.03125	0.03125	0.6379	0.1
128 sec	1	64	0.015625	0.015625	0.635	0.02
256 sec	7	32	0.03125	0.03125	0.6379	0.1

Figure 1: probability distribution of block creation time

4 네트워크 효율성 및 보안성

최소 하나 이상의 분리된 블록이 생성될 확률은? 노드 n 1이 다음 간격에서 하나의 블록을 찾을 확률("tick")은 다음과 같다.

$$P(X = 1) = (1 - P)^{1-1}P = (1 - P)^{0}P = 1 * P = P$$

여기서 P는 모든 스테이크 홀더의 총 확률이다. 코인(stakes)의 큰 분포를 가정 할 때, 두 번째 스테이크 홀더에 대한 확률은 n2가 같은 기간 (다음 간격 이전에)에 또 다른 유효한 블록을 찾도록한다고 말한다. 위에서 본 바와 같이

$$P = p^{\sum_{1}^{s} b_i} = p * B$$

잔고가 *bn1* 인 *n1*이 유효한 블록을 찾았다는 사실을 취하면 다른 블록을 찾을 수 있는 확률은 다음과 같다:

$$P^{\prime} = p^{\sum_{1}^{n_1-1} b_i} + p^{\sum_{n_1+1}^{s} b_i} = p^{\sum_{1}^{s} b_i} + p * b_{n1} + p^{\sum_{n_1+1}^{s} b_i - p * b_{n1}} =$$

$$= p^{\sum_{1}^{s}} - p * b_{n1} = p * B - p * b_{n1} = P - p * b_{n1} \approx P$$

그 이후로 $\frac{b_{n1}}{B} \approx 0$ (우리는 스테이크홀더에게 큰 분포를 가정했다). 요컨대, 고아 차단이 형성될 확률 P_{or} 는

$$P_{or} = P$$

$$* P^{\prime} \approx P * P = P^2 \qquad \approx {}_k 2 =$$

Example: 64초의 블록 생성과 입도의 목표 $g = 7$ $_2$

$k = \frac{target}{} = \frac{64}{} = 8$ $_{g+1}$이므로 $_{7+1}P = \frac{1}{} = {}_1$ 이고 $\frac{1}{k}$ P_8 $or \approx P_2 = \frac{1}{8^2} = \frac{1}{64} =$ 1.5625% 따라서 고아 블록은 그다지 많지 않다. 64 개 중 고아 블록 하나만 $k = 8$이면 생성된다. 그러나 $k = 4$이면 어떻게 될까?

$\frac{1}{4^2} = \frac{1}{16} = 6.25\%$

왜 고아 블록이의 퍼센트에 관심을 가져야 하는지 의문이 생긴다. 그러나 잠시동안 고아 블록이 없다고 해보자. 스테이크홀더가 체인을 성공적으로 공격할 수 있는 방법은? PoS에서 독립체 또는 그룹에 코인의 50% 이상이 있으면 성공적인 공격을 시작하고 블록을 되돌리고 다시 쓸 수 있다. 왜 이런 일이 일어나는지 보는 것은 어렵지 않다. *a는 그의 계정에* B_a 잔액을 가지고 있다. 그는 *n 블록을 되돌리려고 하므로 "돌아가고" 원하는 지점에서 스테이킹을 시작한다.알고리즘이 더 긴 체인을 승인함에 따라 개인 채인은 기록에서 n개의 블록으로 돌아가기 때문에 검증되지 않는다. 그러나 그는 자신의 체인에만 지분을 가지고 있기 때문에 현재 사용 가능한 총 잔액이* $B^{\prime} = B - B_a$ *이므로 나머지 스테이크홀더는 p 확률이 아니* $_P$ 2보다 작다. 여기서 *B는 코인의 총량이다.*

$$B_a > \frac{B}{2} \Leftrightarrow p * B_a > p * \frac{B}{2} \Rightarrow p * B_a > p * B^{\prime} \Leftrightarrow P_a > P_r$$

여기서 P_a 및 P_r 은 각각 공격자와 나머지 네트워크에 대한 블록을 형성할 확률이다. 시간이 지날수록 공격자는 네트워크 나머지 부분보다 더 자주 새로운 블록을 형성할 것이며, 마침내 그의 체인이 원래의 길이를 통과할 것이다. 그래서 그는 *n* 블록을 성공적으로 되돌렸다(다시 작성).
실제로 PoW 합의 프로토콜에서도 가능한 이러한 종류의 공격은 고아 블록이 있는 경우 훨씬 쉬울 수 있다. 네트워크의 효율성을 함수 $ef(o) = 1 - o$, 로 정의하면, 여기서 o 는 고아 블록의 백분율이다. 이전 공격의 한계는 다음과 같다.

$$s(ef) = \frac{ef}{1 + ef}$$

여기서 s는 안전 기능이며 효율성 기능이다. 예를 들어, 고아 블록이 없는 경우,

$$o = 0,\ ef(o) = 1 - o = 1 - 0 = 1$$

$$s(ef) = \frac{ef}{1 + ef} \Leftarrow s(1) = \frac{1}{1 + 1} \Leftrightarrow s(1) = 0.5$$

k	p	Orphaned %	net.ef. (ef)%	safety (”democracy” attack)%
64	0.015625	0.0244140625	99.9755859375	0.49993895739226
32	0.03125	0.09765625	99.90234375	0.499755740107474
16	0.0625	0.390625	99.609375	0.499021526418787
8	0.125	1.5625	98.4375	0.496062992125984
4	0.25	6.25	93.75	0.483870967741936
2	0.5	25	75	0.428571428571429

Table 1: orphaned blocks , network efficiency and safety

따라서 고아 블록이 없으면 공격자는 코인의 50% 이상을 필요로 한다. 그러나 o = 6.25% 의 경우 $ef(0.0625) = 1 \ 0.0625 = 0.9375 = 93.75\%$ 이며 안전 제한은 다음과 같다:

$$s(ef) = \frac{ef}{1 + ef} = \frac{0.9375}{1 + 0.9375} \approx 0.4839 = 48.39\%$$

이제 그 시스템의 안전성이 낮다는 것이 분명하다.

위의 표에서 고아 블록이 체인의 보안과 어떻게 관련되는지 확인할 수 있다.

5 용어

Consensus algorithm(합의 알고리즘): 알고리즘 또는 프로토콜은 분산 시스템이 합의에 도달하는 과정이다. 블록 체인의 경우 노드가 다음 블록을 생성하고 체인에서 트랜잭션을 확인하고 기록하는 방법을 의미한다. **Geometric distribution(기하 분포)**: 이산 확률 분포의 유형

Democracy attack (PoS)(평등 공격 PoS): 다수의 스테이크 소유자가 연쇄 이력을 바꿀 수 있는 공격.

Stake grinding attack(스테이크 그라인딩 공격): 공격자가 계산 능력을 사용하여 성공적인 포크를 만드는 연결 유형. 즉, PoS를 사이드 체인의 많은 해시를 계산하는 PoW로 변환한다

Proof of Stake(지분증명): 스테이킹(자본 리소스)에 기반한 합의 알고리즘 **Proof of Work(작업 증명)**: 많은 계산 (하드웨어 및 전기 자원)을 기반으로 한 합의 알고리즘

참고 문헌

[1] *Gossip protocol* https://en.wikipedia.org/wiki/Gossip_protocol

[2] *Nielsen's Law* http://wiki.p2pfoundation.net/Nielsen’ s_Law

[3] *Kryder's Law* https://en.wikipedia.org/wiki/Mark_Kryder#Kryder’ s_law_projection

[4] *13-th IEEE International Conference on Peer-to-Peer Computing Informa- tion Propagation in the Bitcoin Network* https://www.tik.ee.ethz.ch/file/49318d3f56c1d525aabf7fda78b23fc0/P2P2013_041.pdf
[5] *Christian Walck , Hand-book on STATISTICAL DISTRIBUTIONS for experimentalists* http://www.stat.rice.edu/~dobelman/textfiles/DistributionsHandbook.pdf