The Tangle

Serguei Popov[1], for **Jinn Labs**

2017년 8월 22일. Version 1.2

초록

이 백서에서는 사물인터넷 (IoT) 산업을 위한 가상화폐 iota의 수학적 기반을 분석한다. 이 새로운 가상화폐의 주 특징은, 거래를 저장하는데 블록체인 대신 방향성 비순환 그래프(directed acyclic graph, DAG)인 탱글(tangle)을 사용한다는 것이다. 탱글은 블록체인의 다음 진화적 단계로, 자연적으로 블록체인 기술을 뛰어넘으며, 글로벌 단위의 초소규모 거래를 위해 필요한 기능들을 제공한다.

특히, 이 글의 본질적 공헌들 중 하나는 마르코프 연쇄 몬테카를로(Markov Chain Monte Carlo, MCMC) 알고리즘 종류이다. 이 알고리즘들은 갓 도착한 거래가 탱글에 부착될 site를 선택해준다.

1 시스템의 소개와 설명 (Introduction and Description of the System)

지난 6년동안의 비트코인의 등장과 성공은 블록체인 기술은 실세계의 가치를 가진다는 것을 증명하였다. 하지만, 이 기술은 가상화폐의 일반적인 전세계적 플랫폼으로 사용되는 것을 막는 여러 가지 약점들 또한 가지고 있다. 한가지 주목할만한 단점은 어떠한 가치의 거래에 대한 거래 수수료의 개념이다. 빠르게 개발되는 사물인터넷 산업에서 초소규모 거래의 중요성은 증가할 것이고, 이동하는 가치보다 더 큰 수수료를 지불하는 것은 논리적이지 못하다. 더 나아가, 블록체인 인프라(infrastructure)에서 수수료는 블록 생성자들에게 보상으로 작용하기 때문에 없애기 어렵다. 이것은 기존의 가상화폐 기술의 또 다른 문제, 특히 시스템의 이질적인(heterogeneous) 성질로 이어진다. 블록체인 시스템에는 거래를 요청(issue)하는 자와, 거래를 승인(approve)하는 자, 두 종류의 참여자가 있다. 시스템의 이런 구조는 결국 몇몇 참여자들 간에 차별이 생기게 되며, 모든 구성요소들이 해결에 자원을 소모하게 하는 갈등들을 만들게 된다. 앞서 말한 문제들은 비트코인과 다른 많은 가상화폐들의 기반인 블록체인과 원초적으로 다른 해결책을 찾게 한다.

[1] mthcl(Bitcointalk 가상화폐 포럼 유저)로도 알려져 있다; 필자의 연락 정보: e.monetki@gmail.com

이 백서에서는 블록체인 기술을 이용하지 않는 혁신적인 접근방안에 대해 논의한다. 이 접근방안은 현재 IoT 산업을 위해 특별히 설계된 iota라는 가상화폐에 적용되어 있다. 이 글의 목적은 탱글의 일반적인 특징들에 집중하고, 블록체인 없이 분산된 거래장부를 유지하려 할 때 생기는 문제점들에 대해 논의하는 것이다. Iota 프로토콜의 구체적인 구현방법에 대해서는 논하지 않는다.

일반적으로, 탱글 기반 가상화폐는 다음과 같이 작동한다. 전세계적 블록체인 대신, 탱글이라고 불리는 DAG(directed acyclic graph, 방향성 비순환 그래프)가 있고, 노드(node)에 의해 요청된 거래들은 거래를 저장하는 거래장부인 탱글 그래프의 site 집합을 구성한다. 탱글의 선(edge) 집합은 다음 방법으로 얻어진다: 새로운 거래가 도착하면, 그것은 이전의 거래 2개를[2] 승인해야 한다. 이 승인들은 그림 1에 묘사된 것처럼[3] 방향이 있는 선(directed edge)들로 표현된다. 만약 거래 A와 거래 B간에 방향이 있는 연결선은 없지만, A에서 B까지 길이가 2 이상인 방향이 있는 경로(directed path)가 있으면, 우리는 A가 B를 간접적으로 승인(또는 참고)한다고 한다. 또한, 다른 모든 거래들에 의해 직접 또는 간접적으로 승인되는 "제네시스(genesis)" 거래가 존재한다 (그림 2). 제네시스 거래는 다음과 같이 설명된다. 탱글의 초창기에, 모든 토큰들을 가지고 있었던 주소가 있었고, 이 토큰들은 제네시스 거래를 통해 다른 "창립자"들의 주소들로 송금되었다. 여기서 모든 토큰들은 제네시스 거래에서 생성되었다는 것을 강조한다. 미래에 토큰들은 생성되지 않을 것이며, 채굴자들이 허공으로부터 금융적 보상을 받는 것과 같은 채굴은 없을 것이다.

술어들에 대한 빠른 정리: 사이트들은 탱글 그래프 상에 표현된 거래들이다. 네트워크는 노드들로 구성되어 있는데; 노드들은 거래를 요청하고 검증하는 독립체(entity)들이다.

탱글의 핵심 아이디어는 다음과 같다: 사용자가 거래를 요청하기 위해서는, 다른 거래들을 승인하는데 일해야 한다. 따라서, 거래를 요청하는 사용자들은 네트워크의 보안에 기여한다. 노드들은 승인된 거래가 충돌(conflicting)하지 않는지 확인한다고 가정된다. 만약 노드가 어떤 거래가 탱글의 과거와 충돌한다고 알게 되면, 노드는 직접 또는 간접적으로 그 충돌하는 거래를 승인하지 않을 것이다[4].

거래가 추가적인 승인들을 받을수록, 더 높은 확신도로 시스템에 의해 받아들여진다. 다른 말로, 시스템이 이중지불(double-spending) 거래를 받아들이게 하기 어려울 것이다. 우리는 노드가 승인할 거래를 선택하는 것에 대해 어떠한 규칙도 강요하지 않는다. 대신, 만약 많은 수의 노드들이

---

[2] 이것이 가장 간단한 규칙이다. 거래가 $k \geq 2$에 대해 다른 k개의 거래들을 승인해야 하거나, 아예 다른 규칙을 사용하는 시스템도 연구될 수 있다.

[3] 각 그림에서 시간은 항상 왼쪽에서 오른쪽으로 증가한다.

[4] 만약 노드가 충돌하는 거래를 승인하는 새 거래를 요청한다면, 그 노드는 다른 노드들이 자신의 새 거래를 승인하지 않을 것을 감수한다.

어떤 참조(reference) 규칙을 따른다면, 어떤 고정된 노드에 대해 그것은 같은 종류의 규칙을 따르는 것이 좋을 것이라고 말하는 바이다[5]. 이것은 특히 미리 설치된 펌웨어(firmware)를 가지는 특수화된 칩들이 노드로 작용하는 IoT의 맥락에서 합리적인 가정이라고 생각된다.

거래를 요청하기 위해, 노드는 다음을 하게 된다:

- 노드는 알고리즘에 따라 승인할 다른 두 개의 거래를 선택한다. 일반적으로, 그 두 거래는 일치할 수도 있다.
- 노드는 선택한 두 거래가 충돌하지 않는지 확인하고, 충돌하는 거래는 승인하지 않는다.
- 노드가 유효한 거래를 요청하기 위해서, 노드는 비트코인 블록체인에서와 비슷하게 암호학적 퍼즐을 풀어야 한다. 이것은 승인된 거래의 데이터와 넌스(nonce)의 해시값이 특정한 형태를 가지게 하는 넌스를 찾음으로써 이뤄낸다. 비트코인 프로토콜의 경우에, 해시값은 앞부분의 0의 개수가 정해진 개수 이상이어야 한다.

Iota 네트워크는 비동기적(asynchronous)이기 때문에 일반적으로 노드들은 모두 같은 집합(set)의 거래들을 보고 있지는 않는다. 또한, 탱글에는 충돌하는 거래가 있을 수도 있다. 노드들은 어떤 유효한[6] 거래들이 거래장부에 있을 권리가 있는지 의견 일치(consensus)를 이룰 필요가 없고, 이것은 모든 거래들은 탱글에 있을 수 있다는 것을 뜻한다. 하지만, 충돌하는 거래가 있는 경우에, 노드들은 어떤 거래들을 버릴(orphan)[7] 것인지 결정하여야 한다. 노드들이 두 충돌하는 거래들 중 하나를 고르는데 사용하는 주된 규칙은 다음과 같다: 노드가 끝점 선택 알고리즘(섹션 4.1 참조)을 여러 번 실행한 뒤[8], 두 거래들 중 어느 것이 선택된 끝점으로부터 간접적으로 승인될 확률이 더 높은지 본다. 예를 들어, 어떤 거래가 100번의 끝점 선택 알고리즘 중 97번 선택됐다면, 우리는 그것은 97%의 확신도로 컨펌(confirm) 되었다고 한다.

또한, 다음 물음에([4] 참조) 대해 코멘트(comment)해보자: 무엇이 노드들이 거래들을 확산시키도록 동기부여 하는가? 모든 노드는 이웃으로부터 몇 개의 새로운 거래들을 받았는지 통계를 계산하는데, 만약 어떤 특정한 노드가 너무 게으르다면, 그 노드는 이웃 노드들로부터 버려질 것이다. 따라서, 노드가 거래를 요청하지 않아, 자신의 거래를 승인하는 새 거래들을 공유할 직접적인 인센티브(incentive)는 없더라도 참여에 대한 인센티브는 있다.

섹션 2에서 몇 가지 표기법들을 소개한 뒤, 우리는 승인할 두 거래들 고르는 알고리즘, 거래의

---

[5] 섹션 4.1의 끝부분에서 이것에 대해 더 얘기한다.

[6] 프로토콜에 맞게 요청된 거래들을 뜻한다.

[7] 버려진 거래들은 더 이상 새로운 거래들에 의해 간접적으로 승인되지 않는다.

[8] 위에서 언급하였듯이, 다른 노드들도 동일한 끝점 선택 알고리즘을 따를 것이라고 가정하는 것에는 좋은 이유가 있다.

전체적인 승인도를 측정하는 규칙(섹션 3, 특히 섹션 3.1)과 가능한 공격 시나리오(섹션 4)에 대해 논의할 것이다. 또한, 수학공식을 싫어하는 독자는 각 섹션의 끝에 있는 "결론들"로 바로 건너뛰어도 된다.

[3, 6, 7, 9, 12]에서 볼 수 있듯이, 가상화폐에 DAG를 사용하는 아이디어는 새로운 것이 아니다. 특히, [7]에서는 블록체인이 아닌 트리를 주된 거래장부로 사용하는 개조된 비트코인 프로토콜인 GHOST 프로토콜의 제안을 소개한다. 이런 변화는 컨펌(confirmation) 시간을 감소시키고 네트워크의 전체적인 보안성을 개선시킨다고 보여진다. [9]에서는 DAG기반 가상화폐 모델을 고려한다. 그들의 모델은 다음 이유들로 의해 우리의 모델과 다르다: 그들의 DAG의 사이트들은 독립적인 거래들이 아니라 블록들이고, 시스템의 채굴자들은 거래 수수료를 위해 경쟁하며, 블록 채굴자들에 의해 새 토큰이 생성될 수 있다. 또한 [6]에서 우리와 어느 정도 비슷한 해결책이 제안 되었지만, 끝점 승인 방법들에 대해서는 논의되어 있지 않다. 이 글의 첫 판이 출판된 이후, [8]처럼 DAG 기반 분산된 거래장부를 만들려는 몇 가지 다른 시도들도 나타났었다. 또한, 우리는 개인 대 개인 지불(payment) 채널을 통해 비트코인 초소액 지불(micropayment)이 가능하게 하려는 또 다른 접근법 [2, 10]도 참고한다.

2. 무게와 관련 개념들 (Weights and More)

이 섹션에서는, 거래의 무게와, 관련 개념들을 정의한다. 거래의 무게는 거래를 요청하는 노드가 투자한 일의 양에 비례한다. 현재의 iota에서 무게는 3^n의 값만을 가진다. 이때 n은 양의 정수이며 어떤 비어있지 않은 허용 가능한 값의 구간에 속한다[9]. 사실, 무게가 실제로 어떻게 얻어졌는지는 몰라도 된다. 모든 거래에 어떤 양수, 그 거래의 무게가 달려있다는 것만이 중요하다. 일반적으로, 무게가 더 큰 거래는 무게가 작은 거래보다 더 중요하다. 스패밍(spamming)과 다른 종류의 공격들을 막기 위해 어떠한 자도 짧은 시간 안에 수용 가능한 무게를 가진 거래를 아주 많이 만들어내지 못할 것이라고 생각된다.

우리에게 필요한 개념들 중 하나는 거래의 누적 무게이다: 누적 무게는 거래의 자체 무게와 그 거래를 직접 또는 간접적으로 승인하는 거래들의 무게의 합으로 정의된다. 누적 무게를 계산하는 알고리즘은 그림 1에 나타나 있다. 그림에서 상자들은 거래들을 나타내고, 각 상자의 동남쪽 끝에 있는 작은 숫자들은 거래의 자체 무게, 굵은 숫자들은 누적 무게를 나타낸다. 예를 들어, 거래 F는 거래 A, B, C, E에 의해 직간접적으로 승인되고, 따라서 F의 누적 무게는 F의 자체 무게와 A, B, C, E의 무게의 합인 $9 = 3 + 1 + 3 + 1 + 1$이다.

[9] 이 구간 또한 유한하여야 한다-섹션 4의 "큰 무게 공격"을 보아라.

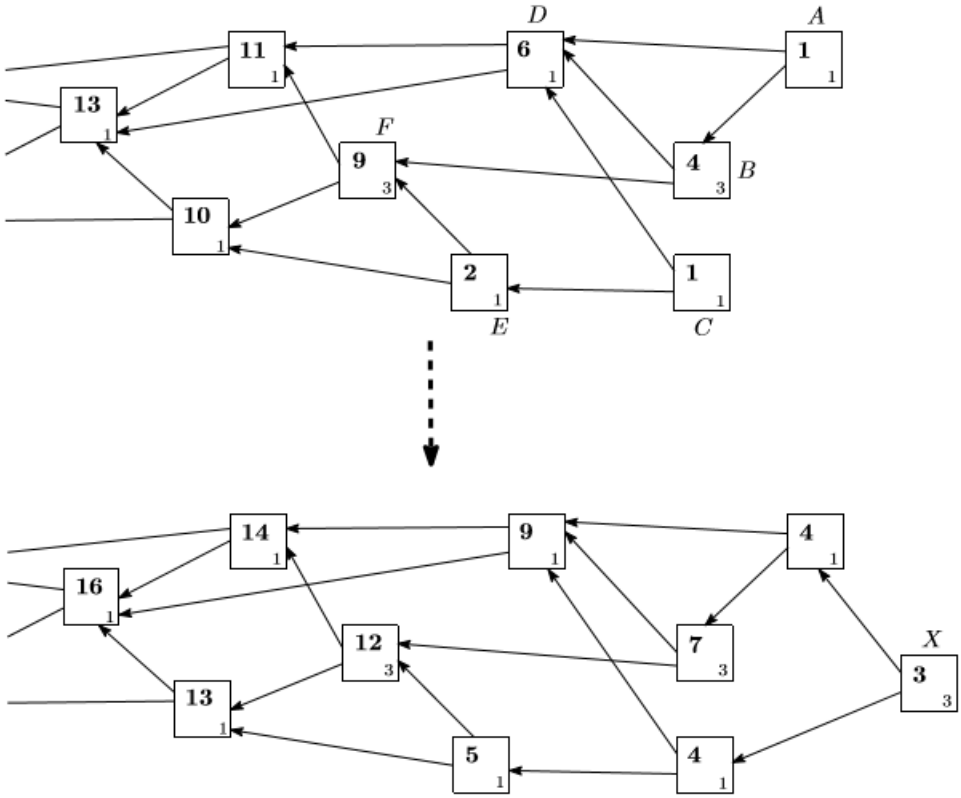

그림 1: 새로 요청된 거래 $\boldsymbol{X}$ 전후의 무게들이 표현된 DAG. 상자들은 거래를, 각 상자의 동남쪽 끝에 있는 작은 숫자는 자체 무게를, 굵은 숫자는 누적 무게를 나타낸다.

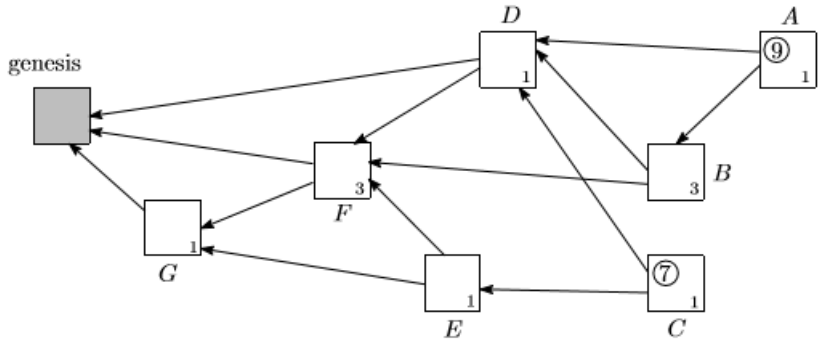

그림 2: 사이트 $\boldsymbol{A}$, $\boldsymbol{C}$의 점수가 계산되었고, 각 사이트에 자체 무게가 지정된 DAG.

탱글 그래프에서 승인되지 않은 거래들을 "끝점"이라고 정의하자. 그림 1의 위쪽의 탱글 스냅샷(snapshot)에서 유일한 끝점들은 A와 C이다. 아래쪽의 탱글 스냅샷에서 새로운 거래 X가 와서 A와 C를 승인하면, X는 유일한 끝점이 된다. 이때 다른 모든 거래들의 누적 무게는 X의 자체 무게

인 3만큼 증가한다.

승인 알고리즘에 대해 논의하기 위해서는 다른 몇 가지 변수를 소개해야 한다. 먼저, 탱글의 거래 site의 변수들에 대해 밑에 소개한다.

- 높이: 제네시스 거래까지의 가장 긴 방향이 있는(oriented) 경로의 길이;
- 깊이: 어떤 끝점까지의 가장 긴 역방향(reverse oriented) 경로의 길이.

예를 들어, 그림 2에서 G는 F, D, B, A의 역방향 경로로 의해 높이 1과 깊이 4를 가지고, D는 높이 3과 깊이 2를 가진다. 또한, 점수의 개념을 소개하자. 거래의 점수는 거래가 승인한 모든 다른 거래들의 자체 무게들의 합에 자신의 자체 무게를 더한 값이다. 그림 2에서, 유일한 끝점들은 A와 C인데, 거래 A는 직접 또는 간접적으로 거래 B, D, F, G를 승인하므로, A의 점수는 $1+3+1+3+1=9$이다. 비슷하게, C의 점수는 $1+1+1+3+1=7$이다

이 글에 나오는 주장들을 이해하기 위해서, 모든 거래들의 자체 무게는 1이라고 가정하자. 이 가정하에, 거래 X의 누적 무게는 X를 직간접적으로 승인하는 거래들의 개수에 1을 더한 값이 되고, 점수는 X가 직간접적으로 승인하는 거래들의 개수에 1을 더한 값이 된다.

높이, 깊이, 그리고 점수도 잠깐 논의될 것이지만, 누적 무게가 가장 중요한 값(metric)이라는 것을 알자.

3. 시스템의 안정도와 컷셋들 (Stability of the System, and Cutsets)

$L(t)$를 시간 t에 시스템에 있는 끝점들의 총 개수라고 하고, 이때 확률 과정(stochastic process) $L(t)$의 값은 안정적일 것이라고 예상된다[10]. 더 정확하게, 이 과정은 양재귀(positive recurrent)적일 것이라고 예상된다, 정식 정의들을 위해서는 [11]의 섹션 4.4와 6.5를 보아라. 특히 양재귀적이라는 것은 $t\to\infty$일때 $\mathbb{P}[L(t)=k]$의 극한값이 존재하고 이 극한값은 모든 $k\geq 1$에 대해 양수여야 한다는 것을 의미한다. 따라서 우리는 $L(t)$가 일정한 값을 중심으로 변동하고 무한대로 가지 않을 것이라고 예상한다. 만약 $L(t)$가 무한대로 가게 된다면, 많은 미승인된 거래들이 남겨지게 될 것이다.

$L(t)$의 안정도 특성들을 분석하기 위해서, 우리는 몇 가지 가정을 해야 한다. 한가지 가정은 들어오는 거래들은 많은 수의 거의 독립적인 노드들에 의해 요청되기 때문에, 푸아송 점 과정(Poisson point process)으로 모델링하는 것이다 ([11]의 섹션 5.3 참조). 이 푸아송 과정(Poisson

[10] 과정이 시간-동질적(homogeneous)이라는 추가적인 가정 하에 말이다.

process)에서 거래의 도착률을 λ라고 하자. 간단하게 하기 위해, 이 도착률은 시간에 따라 일정하고, 모든 기기들은 거의 비슷한 전산능력을 갖고 있다고 가정하자. 그 다음, 끝점의 수는 L, 거래의 총 개수는 N일 때 일반적인 기기가 거래를 요청하기 위한 계산을 하는데 걸리는 평균 시간을 $h(L,N)$이라고 하자. 이제 모든 노드들이 다음과 같이 행동한다고 가정하자: 거래를 요청하기 위해, 노드는 두 개의 끝점을 무작위로 선택한 뒤, 승인한다. 일반적으로 이 방식은 실용적으로는 여러 단점들을 가지기 때문에 정직한 노드들이 이 방식을 사용하는 것은 좋지 않다. 특히, 이 방식은 게으르거나 악의적인 노드로부터 충분한 보호를 해주지 못한다 (밑의 섹션 4.1을 보아라). 하지만 이 모델은 분석하기가 쉽고, 더 복잡한 끝점 선택 방법에 대한 시스템의 행동에 관해 통찰력을 기를 수 있기 때문에 여기서 계속 사용할 것이다. 푸아송 과정의 각 현상들을 가능한 유형들로 독립적으로 분류한다면 각 유형에 대해 현상의 푸아송 과정들은 서로 독립적인 푸아송 과정들로 취급 가능하다는 [11]의 명제 5.3으로 의해, 이 간단한 끝점 선택 방법에서 각 끝점에 대한 승인들의 푸아송 과정은 독립적이고 $2\lambda/L$의 rate을 가진다고 할 수 있다. 따라서 시간 $h(L,N)$동안 아무도 주어진 끝점을 승인하지 않을 확률은 다음과 같다.

$$\mathbb{P}\left[\textit{시간 } h(L,N) \textit{ 동안 아무도 주어진 끝점을 승인하지 않음}\right] = \exp\left(-\frac{2\lambda h(L,N)}{L}\right) \qquad (1)$$

이것은 우리의 기기가 거래를 요청한 시각에 끝점들의 총 개수의 증가량의 기댓값은 다음과 같다는 것을 뜻한다.

$$1 - 2\exp\left(-\frac{2\lambda h(L,N)}{L}\right) \qquad (2)$$

위 공식에서, "1"은 우리의 거래에 의해 생성된 새 끝점에 대응되고, 두 번째 항은 승인되어 없어진 끝점들의 수의 기댓값에 대응된다. (1)에 따르면, 처음 승인을 위해 선택된 각각의 끝점은[11] $\exp\left(-\frac{2\lambda h(L,N)}{L}\right)$의 확률로 거래가 요청된 순간에도 계속 끝점으로 남아있는다. $L(t)$은 사실 가장 가까운 이웃들 간의 전이(nearest-neighbor transition)를 가지는 $\mathbb{N} = \{1,2,3,\ldots\}$에 대한 연속 시간 무작위 행보(continuous-time random walk)이다. 만약 선택된 두 거래가 이미 승인되었다면, 과정은 한 단위(unit)만큼 우측으로 이동하고, 선택된 두 거래가 전에 승인되지 않았다면, 과정은 한 단위만큼 좌측으로 이동한다. 가능한 마지막 경우에, 과정은 그대로 있는다.

과정의 일반적인 행동을 이해하기 위해서, (2)의 drift[12]는 작은 $L(t)$에 대해서는 양수, 큰 L에 대해서는 음수 값을 가지는 것을 보라[13]. 다른 말로, $L(t)$가 작을 때, drift는 증가하는 경향이 있

[11] 노드는 항상 적어도 두 개의 끝점이 있으면 두 별개의 끝점을 승인한다는 가정 하에 말이다.

[12] 끝점들의 개수의 증가량의 기댓값.

[13] L이 클 때 이것은 $L \to \infty$여서 $h(L,N) = o(L)$일 경우에 성립하거나, 끝점의 선택방식이 전산, 확산 시간에 별로 영향을 주지 않을 때 성립한다.

고, $L(t)$가 클 때, drift는 감소하는 경향이 있다. 따라서, $L(t)$의 "전형적인" 값 L_0는, (2)의 값이 0이 되어, 끝점들의 수가 증가 또는 감소하려는 경향이 없을 때 나타난다. L_0의 값은 다음과 같다.

$$L_0 \approx \frac{2\lambda h(L_0, N)}{\ln 2} \approx 2.89 \cdot \lambda h(L_0, N) \quad (3)$$

위에서 정의된 L_0가 끝점 집합의 일반적인 크기이다. 또한, 거래가 처음으로 승인되기까지 걸리는 시간의 기댓값은 약 $L_0/2\lambda$이다.

어떤 고정된 시간 t에 대해 어떤 순간 $s \in [t, t + h(L_0, N)]$에 끝점이었던 거래들은 보통 컷셋을 이룬다[14]. 시간 $t' > t$에 요청되었던 거래로부터 제네시스 거래까지의 경로는 무조건 이 컷셋을 지나야 한다. 가끔씩 탱글의 새로운 컷셋의 크기가 작아지는 것은 중요하다. 작은 컷셋들은 DAG 간략화와 다른 일(task)들에 체크포인트로 사용될 수 있다.

위의 "순수 무작위적" 승인 방법은 끝점을 승인하는 행위를 장려하지 않기 때문에 실제로는 별로 좋지 않은 방법이다. "게으른" 사용자가 항상 아주 오래된 거래들 중 고정 쌍만을 승인하여 처벌을 받지 않으면서도 더 최근의 거래들의 승인에 기여하지 않을 수도 있다[15]. 또한, 악의적인 자가 고정된 쌍의 거래들을 승인하는 많은 수의 거래들을 생산해내, 끝점 수를 인위적으로 증가시킬 수도 있다. 이러면 나중의 거래들이 이 끝점들을 높은 확률로 선택하게 하는 것이 가능해져, 정직한 노드들의 끝점들이 버려지게 할 수 있다. 이러한 문제들을 방지하기 위해, 더 나은 끝점들을 더 추구하는 방법을 사용해야 한다. 이러한 방법들의 예들 중 하나는 아래 섹션 4.1에 나와있다[16].

어떤 거래가 첫 승인을 받기까지 걸리는 시간의 기댓값에 대한 논의를 시작하기 전에 우리는 먼저 두 가지 상황을 분별할 수 있다 (그림 3).

- 저부하(low load): 끝점의 전형적 개수가 작고, 자주 1이 된다. 이것은 새로 들어오는 거래들이 너무 적어 서로 다른 여러 개의 거래들이 동일한 끝점을 승인할 확률이 낮을 때 일어날 수 있다. 또한, 네트워크의 지연이 아주 적고 기기들의 전산속도가 빠르다면, 많은 끝점들이 나타나기 어렵다. 이것은 새로운 거래의 수가 어느 정도 많을 때도 가능하다. 또한, 끝점들의 수를 인위적으로 증가시키려는 공격자들이 없다고 가정해야 한다.

---

[14] 적어도 노드들이 끝점들을 승인하려고 시도하는 경우에 말이다.

[15] 우리는 어떤 특정 끝점 선택 방법도 강요하지 않는다고 강조한다. 공격자는 자신이 편한대로 끝점을 선택할 수 있다.

[16] 공격 방법들은 시스템의 요소에 숨어있기 때문에 끝점 선택 방법이 탱글 기반 가상화폐의 건설에서 가장 중요한 요소라는 것이 필자의 생각이다. 또한, 보통 특정한 끝점 선택 방법을 강요할 수 있는 방법이 없기에, 노드들이 자발적으로 공통의 방법을 따르는 것에는 좋은 이유가 있어야 한다. 가능한 이유 중 하나는 노드가 충분히 많은 수의 다른 노드들이 동일한 끝점 선택 방법을 사용한다는 것을 알고 동일한 방법을 따라하는 것이다.

- 고부하(high load): 끝점의 일반적인 개수가 많다. 이것은 새로운 거래의 수가 많고, 전산의 지연과 함께 네트워크 지연이 일어나면 서로 다른 여러 개의 거래들이 같은 끝점을 승인할 확률이 높아진다.

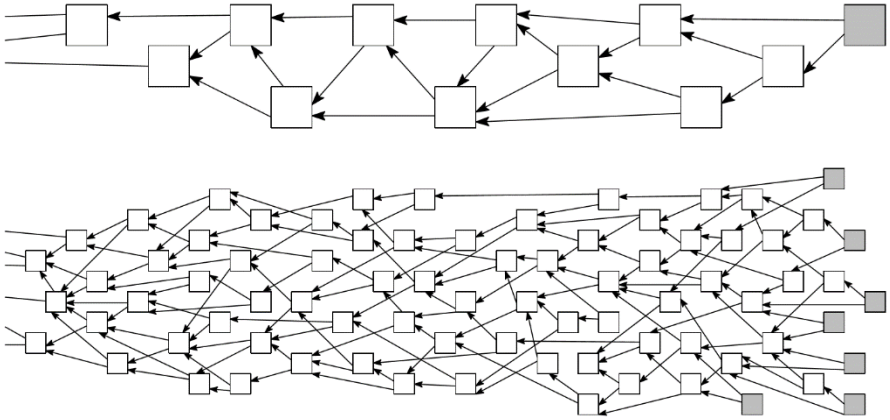

그림 3: 들어오는 거래의 저부하(위)와 고부하(아래) 상태. 흰색 상자는 검증된 site를, 회색 상자는 끝점을 나타낸다.

이 분별은 비공식적이며, 두 상황 사이에 뚜렷한 경계는 존재하지 않는다. 그래도 이 두 극한의 상황을 고려하는 것은 유익할 수 있다.

저부하의 상황일 때는 비교적 간단하다. 수가 적은 새로 들어오는 거래들 중 하나가 빠른 시간 내에 주어진 끝점을 승인하게 되기 때문에 첫 승인은 λ^{-1}에 비례하는 시간 이후에 일어나게 된다..

이제 L_0의 값이 큰 고부하의 상황을 고려하자. 위에서 말한 것처럼, 서로 다른 끝점들의 승인들의 푸아송 과정은 서로 독립적이고 $2\lambda/L_0$의 rate을 가진다고 가정할 수 있다. 따라서, 어떤 거래가 첫 승인을 받을 때까지 걸리는 시간의 기댓값은 $L_0/(2\lambda) \approx 1.45h$ 정도이다.

하지만, 더 정교한 승인 방법들에[17] 대해서는 거래가 다른 거래들에 의해 승인될 때까지 오랫동안 수동적으로 기다리는 것은 좋은 방법이 아니다. 이것은 더 승인이 잘되는 "더 좋은" 끝점들이 계속 나타나기 때문이다. 대신, 거래가 $L_0/2\lambda$보다 훨씬 긴 시간 동안 승인을 기다리고 있었다면, 추가적인 빈 거래를[18] 통해 이 거래를 홍보하는 것이 좋은 방법이다. 다른 말로, 노드는 자신

[17] Iota의 미래 버전들에서 "더 높은" 질의 끝점들을 추구하는 방법들

[18] 어떤 토큰 이동도 없지만, 다른 두 거래를 승인해야 하는 거래이다. 빈 거래의 생성은 네트워크의 보안에 기여한다.

의 전 거래와 "더 좋은" 끝점들 중 하나를 승인하는 빈 거래를 요청하여 빈 거래가 승인 받을 확률을 높일 수 있다.

섹션 4.1에서 볼 수 있듯이, 높이와 점수에 기반하는 승인 방법은 특정 공격에 취약할 수 있다. 그 섹션에서 우리는 그런 공격으로부터 방어할 수 있는 더 정교한 끝점 선택 방법들을[19] 논의할 것이다. 그 전까지, 새 거래가 무작위로 선택한 끝점 두개를 승인하는 끝점 선택 방법은 계속 고려할 만하다. 이 방법은 가장 분석하기 쉽고, 따라서 이 방법을 통해 탱글의 정량적, 정성적 행동에 대한 통찰을 키울 수 있다.

결론들:

1. 우리는 저부하와 고부하의 두 가지 상황을 분별하였다 (그림 3).
2. 저부하의 상황에서는 적은 수의 끝점들이 존재한다. λ가 새로운 거래들이 유입되는 rate 일 때, 끝점은 $\Theta(\lambda^{-1})$의 시간 이후에 첫 승인을 받는다.
3. 고부하의 상황에서 끝점의 전형적인 개수는 새 거래가 사용하는 끝점 승인 방법에 의존한다.
4. 만약 거래가 무작위적으로 두 개의 끝점을 골라 승인하는 방법을 사용한다면, 끝점의 전형적인 개수는 (3)으로 나타난다. 끝점의 전형적인 개수에 대해서는 이 방법이 가장 최적이다. 하지만, 이 방법은 끝점을 승인하는 행위를 장려하지 않기 때문에 실제로 사용하기에는 실용적이지 못하다.
5. 공격들과 다른 네트워크 문제들을 해결하기 위해서는 더 정교한 방법들이 필요하다. 섹션 4.1에서 이런 방법들이 논의된다.
6. 고부하의 상황에서, h가 노드가 전산/확산에 걸리는 평균 시간일 때, 끝점이 승인받기까지의 전형적인 시간은 $\Theta(h)$이다. 하지만, 만약 위 시간 안에 첫 승인이 이루어지지 않는다면, 요청자와/또는 거래를 받는 자는 추가적인 빈 거래를 통해 그 거래를 홍보하는 것이 좋은 방법이다.

[19] 사실, 필자의 생각은 탱글 기반 가상화폐의 가장 중요한 요소는 끝점 선택 방법이라는 것이다. 여기가 바로 많은 공격 방법들이 숨어있는 곳이다. 또한, 보통 특정한 끝점 선택 방법을 강요할 수 있는 방법이 없기 때문에, 어떤 노드가 충분한 수의 다른 노드들이 어떤 끝점 선택 방법을 따른다는 것을 안다면, 그 노드도 자발적으로 그 방법을 따르도록 선택할 것이다.

3.1 누적 무게는 보통 얼마나 빨리 커지는가?

(How fast does the cumulative weight typically grow?)

네트워크가 저부하 상태에 있을 때, 어떤 거래가 여러 번 승인된 이후에는 모든 신규 거래들이 그 거래를 참고하게 되기 때문에 그 거래의 누적 무게는 λ의 속도로 커진다[20].

네트워크가 고부하 상태에 있는 경우, 큰 누적 무게를 가지는 오래된 거래는 모든 신규 거래들이 그 거래를 참고하기 때문에 속도 λ로 누적무게가 증가한다. 반면에 거래가 탱글에 처음 추가되었을 때는 승인을 받기까지 좀 시간이 걸릴 수도 있다. 이 시간간격 동안, 거래의 누적 무게는 무작위적 특성을 가진다. 거래가 몇번의 승인을 받은 이후의 누적 무게 증가속도를 계산하기 위해서, 시간 t에서의 누적 무게의 기댓값을 $H(t)$, 그리고 시간[21] t에 거래를 승인하는 끝점들의 개수의 기댓값을 $K(t)$라고 정의하고, $h \coloneqq h(L_0, N)$으로 줄여서 사용하자. 간단히 표현하기 위해 끝점들의 수는 시간에 따라 거의 일정하게 L_0로 유지된다고 가정하자. 여기서는 "두 무작위적으로 선택된 끝점을 승인"하는 방법에 대해 풀지만, 다른 합리적인 방법들에 대해서도 정성적 특성은 비슷할 것이라고 예상된다.

노드는 거래를 요청하기 전에 무조건 어떤 양의 연산과 검증을 해야 하기 때문에 네트워크에 진입하는 거래는 보통 시간 $t-h$의 시스템의 상태에 기반하여 승인할 두 끝점을 고른다. 그 거래가 탱글에 있는 "우리의" 끝점들 중 적어도 하나를 승인할 확률은 $1-\left(1-\frac{K(t-h)}{L_0}\right)^2 = \frac{K(t-h)}{L_0}\left(2-\frac{K(t-h)}{L_0}\right)$다[22]. [11]의 예제 6.4처럼, $\delta > 0$가 작은 경우에 대해서는 다음과 같이 쓸 수 있다.

$$H(t+\delta) = H(t) + \lambda\delta\frac{K(t-h)}{L_0}\left(2-\frac{K(t-h)}{L_0}\right) + o(\delta)$$

따라서 다음 미분방정식을 얻을 수 있다.

$$\frac{dH(t)}{dt} = \lambda\frac{K(t-h)}{L_0}\left(2-\frac{K(t-h)}{L_0}\right) \tag{4}$$

(4)를 이용하기 위해서는 먼저 $K(t)$를 계산해야 한다. 이것은 쉽지 않은데, 시간 $t-h$에서의 끝점은 시간 t에서는 끝점이 아닐 수도 있고, 새로운 거래가 이런 끝점을 승인하게 되면 원래의 거래를 승인하는 끝점들의 전체 개수가 1만큼 증가하기 때문이다. 시간 $t-h$에서의 끝점이 시간 t에도 끝점으로 남아있을 확률은 대략 $1/2$이다. 이것을 검증하기 위해서는, (3)의 식을 (1)에 대입해

[20] 모든 거래들의 자체 무게는 1이라고 가정하였다는 것을 기억하고, 따라서 누적 무게는 거래를 직접 또는 간접적으로 승인하는 거래들의 개수에 1을 더한 값이다.

[21] 단순함을 위해, 우리는 우리의 거래가 생성된 순간부터 시간을 잰다.

[22] 좌변의 식은 승인된 두 끝점이 우리의 것이 아닐 확률을 1에서 뺀 값이다.

보라. 따라서, 시간 t에는 $K(t-h)$의 절반 정도의 끝점들이 미승인된 끝점으로 남아있고, 나머지 절반은 적어도 한번의 승인을 받았을 것이다. 시간 t에도 계속 끝점으로 남아있는 시간 $t-h$에서의 $K(t-h)/2$개의 끝점 집합을 A, 시간 t전에 승인을 받은 남은 $K(t-h)/2$개의 끝점 집합을 B라고 하자. 그 다음, 새 거래가 B에서 적어도 1개의 거래를 승인하고, A의 끝점은 승인하지 않을 확률을 p_1, 승인 받은 거래 두 개 다 A에 속할 확률을 p_2라고 하자. 다른 말로, p_1과 p_2는 각각 새로운 거래가 왔을 때 "우리의" 끝점의 개수가 1만큼 증가하거나 감소할 확률이다. p_1과 p_2는 다음과 같은 값을 가진다.

$$p_1 = \left(\frac{K(t-h)}{2L_0}\right)^2 + 2 \times \frac{K(t-h)}{2L_0}\left(1 - \frac{K(t-h)}{L_0}\right),$$

$$p_2 = \left(\frac{K(t-h)}{2L_0}\right)^2$$

첫 번째 식을 얻기 위해, p_1의 값은 승인된 끝점 둘 다 B에 속할 확률과 하나의 끝점은 B에 속하고 나머지 하나는 $A \cup B$에 속하지 않을 확률의 두 배의 합이라는 것을 이용하자. (4)와 비슷하게, $K(t)$의 미분방정식은 다음과 같이 나온다.

$$\frac{dK(t)}{dt} = (p_1 - p_2)\lambda = \frac{K(t-h)}{L_0}\left(1 - \frac{K(t-h)}{L_0}\right) \tag{5}$$

(5)를 정확히 풀기는 어려우므로, 단순화를 위해 몇 가지 추가적인 가정을 하자. 먼저, 고정된 $\varepsilon > 0$에 대해 $K(t)$의 값이 εL_0에 도달할 때 그 값은 $(1-\varepsilon)L_0$로 아주 빠르게 커지는 것을 보자. 이제 L_0에 비해 $K(t)$의 값이 아주 작다면, 우리는 (5)의 우변에 있는 마지막 항을[2 3] 버릴 수 있다. $\frac{\lambda h}{L_0} = \frac{1}{2}\ln 2$임을 통해 다음과 같은 (5)의 간단화된 식을 얻고, 이 식은 $K(0) = 1$의 경계조건을 가진다.

$$\frac{dK(t)}{dt} \approx \frac{\ln 2}{2h} K(t-h) \tag{6}$$

우리는 $K(t) = \exp\left(c\frac{t}{h}\right)$ 형태의 해를 찾으므로, 이 형태를 (6)에 대입하면 우리는 다음 식을 얻게 된다.

$$\frac{c}{h}\exp\left(c\frac{t}{h}\right) \approx \frac{\ln 2}{2h}\exp\left(c\frac{t}{h} - c\right)$$

따라서 다음 식을 얻을 수 있다.

$$K(t) = \exp\left(W\left(\frac{1}{2}\ln 2\right)\frac{t}{h}\right) \approx \exp\left(0.27\frac{t}{h}\right) \tag{7}$$

[2 3] 이 값은 1에 가까운 상수일 것이므로, 우변은 $\lambda\frac{K(t-h)}{L_0}$이다.

이것은 대략적인 해인데, 이때 $W(\cdot)$은 람베르트 W-함수(Lambert W-function)라고[24] 불린다. (7)의 양변에 로그를 취하면, $K(t)$의 값이 εL_0에 도달하는 시간은 대략 다음과 같이 나온다.

$$t_0 \approx \frac{h}{\frac{1}{2}\ln 2} \times (\ln L_0 - \ln \varepsilon^{-1}) \lesssim 3.76 \cdot \ln L_0 \tag{8}$$

(4)에서 우변의 마지막 항을 버리면, "적응 기간"(t_0가 (8)과 같을 때 $t \le t_0$)동안 다음이 성립하고,

$$\begin{aligned}\frac{dH(t)}{dt} &\approx \frac{2\lambda}{L_0} K(t-h) \\ &\approx \frac{\ln 2}{h\ \exp\left(W\left(\frac{1}{2}\ln 2\right)\right)} \exp\left(W\left(\frac{1}{2}ln\,2\right)\frac{t}{h}\right) \\ &= \frac{W\left(\frac{1}{2}\ln 2\right)}{h} \exp\left(W\left(\frac{1}{2}ln\,2\right)\frac{t}{h}\right)\end{aligned}$$

따라서 다음이 성립한다.

$$H(t) \approx \exp\left(W\left(\frac{1}{2}ln\,2\right)\frac{t}{h}\right) \approx \left(0.27\frac{t}{h}\right) \tag{9}$$

적응기간 동안 누적 무게 $H(t)$는 일정한 속도 λ로 커진다는 것을 알자. 여기서 우리는 (9)의 "지수함수적 성장"은 적응기간 동안에 누적 무게가 아주 빠르게 성장한다는 것이 아니라 그림 4에 나온 것처럼 행동한다는 것을 뜻한다고 강조한다.

결론들:

1. 저부하 상태에서 거래가 여러 번 승인된 후에, 그 거래의 누적 무게는 일반적인 거래의 평균 무게가 w일때, λw의 속도로 커질 것이다.
2. 고부하의 상황에서는 서로 다른 두 가지의 성장 단계가 있다. 먼저, (9)에 따르면 적응기간 동안 거래의 누적 무게 $H(t)$는 증가하는 속도로 커진다. 적응기간 이후, 누적무게는 λw의 속도로 커진다 (그림 4). 사실 모든 합리적인 방법에 대해 적응기간 이후 모든 새로운 거래들이 거래를 직접 또는 간접적으로 승인할 것이기 때문에 누적 무게는 이 속도로 커질 것이다.
3. 적응기간은 현재 끝점들의 대부분이 간접적으로 거래를 승인할 때까지 걸리는 시간으로 생각할 수 있다. 적응기간의 전형적인 길이는 (8)과 같다.

[24] 오메가 함수(omega function) 또는 product logarithm으로도 알려져 있다; $x \in [0, +\infty)$에 대해 이것은 $x = W(x)\exp(W(x))$의 관계로 특징지어진다.

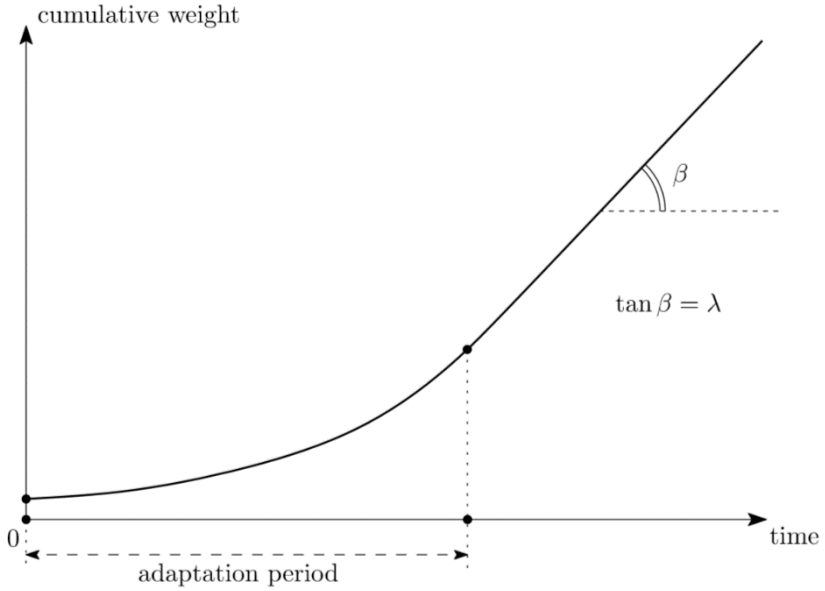

그림 4: 고부하 상태에서의 누적 무게-시간 그래프

4. 가능한 공격 시나리오 (Possible attack Scenarios)

공격자가 혼자 네트워크를 앞지르는 것을 시도하는 공격 시나리오를 논의해보자:

1. 공격자가 상인에게 돈을 송금하고 상인이 거래가 충분한 누적무게를 가졌다고 결정한 이후, 상품을 받는다.
2. 공격자가 이중지불 거래를 요청한다.
3. 공격자가 자신의 전산능력을 사용하여 이중지불 거래는 승인하지만, 상인에게 보낸 원래의 거래는 직접 또는 간접적으로 승인하지 않는 작은 거래들을 많이 요청한다.
4. 공격자는 끝점들을 승인하지 않아도 되는 많은 수의 시빌 정체성들(Sybil identities)를 가지는 것이 가능하다.
5. 3에 대한 또 다른 방법은 공격자가 자신의 모든 전산능력을 사용하여 큰 이중 지불 거래를 요청하는 것이다. 이 거래는 아주 큰 자체 무게를 가지고[25], 상인에게 돈을 지불하기 위해 사용된 진짜 거래 이전의 거래들을 승인한다.

[25] 우리는 여기서 거래의 자체 무게는 변할 수 있다고 가정한다. 밑의 논의에서 자체 무게가 변할 수 있게 놔두는 것이 왜 좋은지 분명해질 것이다.

6. 공격자는 자신의 부정직한 부분탱글이 정직한 부분탱글을 앞지르기를 기대한다. 이것이 일어나게 되면, 주 탱글은 이중 지불 거래로부터 성장하기 시작하고, 상인에게의 거래를 가진 진짜 가지는 버려지게 된다 (그림 5).

사실, 하나의 큰 이중지불 거래를 만드는 방법은 공격자가 성공할 확률을 높인다. 이 수학적 모델의 이상적인 상황에서, 이 공격은 항상 성공한다.

이중지불 거래에게 적어도 3^n의 무게를 주는 넌스를 얻기까지 걸리는 시간을 $W(n)$이라고 놓자. μ가 공격자의 전산능력일 때, $W^{(n)}$은 매개변수[26] $\mu 3^{-n}$을 가지는 지수분포를 따르는 확률변수(exponentially distributed random variable)라고 가정될 수 있다.

상인은 진짜 거래의 누적 무게가 w_0이상일 때 거래를 인정한다고 하고, 이것은 원래 거래로부터 t_0의 시간 단위 이후에 일어난다고 가정하자. 정직한 노드들에 의해 생성된 거래들의 도착률이 λ, 일반적인 거래의 평균 무게는 w일때, 누적 무게는 일정한 속도 λw로 커질 것으로 예상된다. 그 시각 진짜 가지의 전형적인 전체 무게는 $w_1 = \lambda w t_0$이다.

x보다 크거나 같은 가장 작은 정수를 $\lceil x \rceil$라고 하고, $3^{n_0} \geq w_1$이도록 $n_0 = \left\lceil \frac{\ln w_1}{\ln 3} \right\rceil$을 정의하자[27]. 만약 공격자가 t_0의 시간 안에 이중지불 거래에 3^{n_0}이상의 무게를 주는 넌스를 얻는다면, 공격은 성공한다. 이 일이 일어날 확률은 다음과 같다.

$$\mathbb{P}\left[W^{(n_0)} < t_0\right] = 1 - \exp(-t_0 \mu 3^{-n_0}) \approx 1 - \exp(-t_0 \mu {w_1}^{-1}) \approx \frac{t_0 \mu}{w_1}$$

이 근사는 $\frac{t_0 \mu}{w_1}$이 작을 때 성립하며, 이것은 합리적인 가정이다. 만약 이 "즉각적인" 공격이 실패한다면, 공격자는 $n > n_0$에 대해 3^n의 무게를 주는 넌스를 계속 찾는 것을 시도하고, 그것을 찾는 순간 진짜 가지의 전체 무게가 3^n보다 작기를 바랄 수 있다. 이 일이 일어날 확률은 다음과 같다.

$$\mathbb{P}\left[\lambda w W^{(n)} < 3^n\right] = 1 - \exp\left(-\mu 3^{-n_0} \times \left(\frac{3^{n_0}}{\lambda w}\right)\right) = 1 - \exp\left(-\frac{\mu}{\lambda w}\right) \approx \frac{\mu}{\lambda w}$$

보통 $\frac{\mu}{\lambda w}$는 작은 수이지만, 각 "단계" n마다 공격은 일정한 성공 확률을 가진다. 따라서, 공격은 거의 항상 성공할 것이다. 공격이 성공하기까지의 시간은 보통 $3^{\frac{\lambda w}{\mu}}$정도이다. 이 수는 아주 클 수 있지만, "첫 번째"[28] 공격이 성공할 확률은 무시할 수 없고, 따라서 우리는 대책들이 필요하다. 하나의 대책은 위의 자체 무게에 제한을 두거나, 자체 무게를 상수로 고정하는 것이다. 섹션 3에서 말했듯이, 후자 대안은 스패밍으로부터 충분히 방어할 수 없기 때문에 최상의 해결책은 아니다.

---

[26] $\mu^{-1}3^n$의 기댓값을 가진다.

[27] 사실, w_1가 크다면 $3^{n_0} \approx w_1$이다.

[28] 시간 t_0안에.

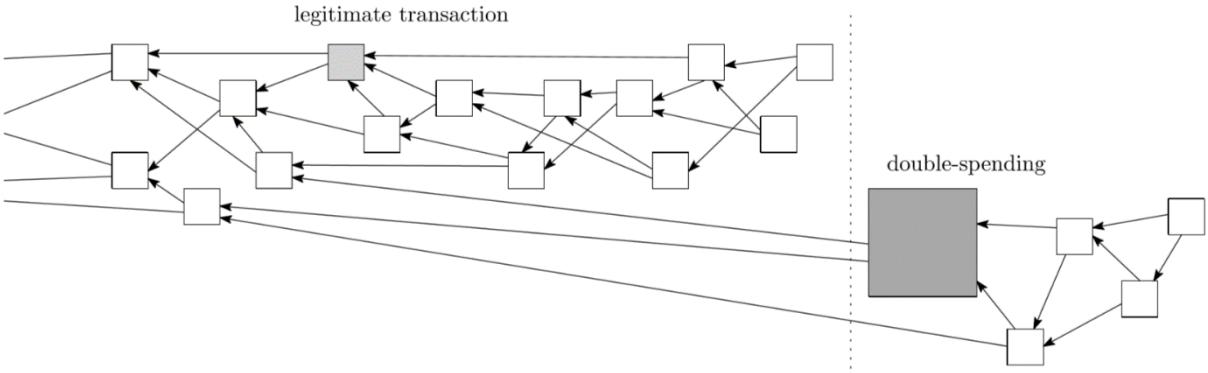

그림 5: "큰 무게" 공격

이제, 자체 무게가 최대 1로 제한된 상황과 이 상황에서 공격의 성공확률에 대해 논해보자.

주어진 거래가 요청되고 시간 t_0후에 누적 무게 w_0를 얻었다고 가정하고, 그 거래의 적응기간이 지났다고 하자. 이 상황에서, 거래의 누적 무게는 속도 λ로 일정하게 커진다. 이제, 공격자가 이 거래에 대해 이중지불을 하려고 한다고 하자. 이중지불을 하기 위해 공격자는 비밀리에 이중지불 거래를 준비하고, 원래의 거래가 상인에게 요청된 시각에[29] 이중지불 거래를 승인하는 무의미한(nonsense) 거래들을 생성하기 시작한다. 상인이 진짜 거래를 받아들인 이후에 공격자의 부분탱글이 진짜 부분탱글을 앞지르게 된다면, 공격자의 이중지불 공격은 성공적이게 된다. 만약 이 일이 일어나지 않는다면, 진짜 거래가 더 큰 누적 무게를 가질 것이고 거의 모든 신규 끝점들이 진짜 거래를 간접적으로 승인할 것이기 때문에 이중지불 거래는 다른 거래들에 의해 승인되지 않을 것이다. 이 경우에 이중지불 거래는 버려지게 될 것이다.

전과 같이, 공격자의 전산능력을 μ라고 하고, 단순화를 위해 거래들은 즉각적으로 확산된다고 가정하자. G_1, G_2, G_3, . . .는 매개변수 μ를[30] 갖는 독립적이고 동일한 분포를 갖는(independent and identically distributed) 지수확률변수(exponential random variables)라고 놓고, $V_k = \mu G_k$, $k \geq 1$를 정의한다. V_1, V_2, V_3, . . .도 매개변수 1을 갖는 독립적이고 동일한 분포를 갖는 지수확률변수이다.

시간 t_0에 상인은 누적 무게 w_0를 가지는 거래를 받아들이기로 결정한다고 가정하자. 공격자가 성공적으로 이중지불을 할 확률을 예상해보자. 매개변수가 1인 지수분포(exponential distribution)의 적률생성함수(moment generating function)를 $M(\theta) = (1-\theta)^{-1}$이라고 놓자 ([14]의 섹션 7.7).

[29] 또는 더 이전에; 이것에 대해서는 나중에 논의한다.

[30] 기댓값 $1/\mu$를 가진다.

$\alpha \in (0,1)$에 대해 다음이 성립한다고 알려져 있다[31].

$$\mathbb{P}\left[\sum_{k=1}^{n} V_k \le \alpha n\right] \approx \exp\bigl(-n\varphi(\alpha)\bigr) \tag{10}$$

여기서 $\varphi(\alpha) = -\ln\alpha + \alpha - 1$는 $\ln M(-\theta)$의 르장드르 변환(Legendre transform)이다. 또한 $\alpha \in (0,1)$에 대해 $\varphi(\alpha) > 0$이 성립한다. 매개변수 1을 가지는 지수확률변수의 예상값은 1이라는 것을 기억하자.

$\frac{\mu t_0}{w_0} < 1$이지 않으면 공격자의 부분탱글이 결과적으로 진짜 부분탱글을 앞지를 확률은 1에 가까울 것이기 때문에 $\frac{\mu t_0}{w_0} < 1$이라고 가정하자. 이제 시간 t_0에 w_0보다 큰 무게를 가지려면, 공격자는 t_0의 시간 동안 최대 자체 무게인 m의 무게를 갖는 거래들을 적어도 w_0개 생성할 수 있어야 한다. 따라서 (10)을 사용하여, 시간 t_0에 이중지불 거래의 누적 무게가 더 클 확률은 대략 다음과 같다는 것을 알 수 있다.

$$\begin{aligned}\mathbb{P}\left[\sum_{k=1}^{w_0/m} G_k < t_0\right] &= \mathbb{P}\left[\sum_{k=1}^{w_0} V_k < \mu t_0\right] \\ &= \mathbb{P}\left[\sum_{k=1}^{w_0} V_k < w_0 \times \frac{\mu t_0}{w_0}\right] \\ &\approx \exp\left(-w_0\varphi\left(\frac{\mu t_0}{w_0}\right)\right)\end{aligned} \tag{11}$$

위 확률이 작기 위해서, $\frac{w_0}{m}$는 커야 하고, $\varphi\left(\frac{\mu t_0}{w_0}\right)$는 아주 작으면 안된다.

시간 $t \ge t_0$에 적응기간이 지났다고 가정하였기 때문에 누적 무게는 λ의 속도로 자라 진짜 거래의 무게는 약 $w_0 + \lambda(t - t_0)$이다. (11)처럼, 시간 $t \ge t_0$에 이중지불 거래의 누적 무게가 더 클 확률은 대략 다음과 같다.

$$\exp\left(-\bigl(w_0 + \lambda(t - t_0)\bigr)\varphi\left(\frac{\mu t}{w_0 + \lambda(t - t_0)}\right)\right) \tag{12}$$

적응기간 동안 누적 무게가 커지는 속도는 λ보다 작기 때문에 $\frac{\mu t_0}{w_0} \ge \frac{\mu}{\lambda}$여야 한다. 따라서, 이중지불이 성공적일 확률은 다음의 값에 비례한다.

[31] 이것은 Large Deviation Principle의 결과이다. 최대값의 단순한 유도과정은 General book [13]과, [14]의 섹션 8.5에 있는 명제(proposition) 5.2에 나와있고, 최소값의 (그리 단순하지는 않은) 유도과정은 [5]의 섹션 1.9에 나와있다.

$$\exp\left(-w_0\varphi\left(\max\left(\frac{\mu t_0}{w_0},\frac{\mu}{\lambda}\right)\right)\right) \tag{13}$$

예를 들어, 공격자의 전산능력이 나머지 네트워크의 전산능력보다 조금 작게 $\mu = 2$, $\lambda = 3$이라고 하자. 그리고 시간 12에 거래의 누적 무게는 32라고 가정하자. 그러면 $\max\left(\frac{\mu t_0}{w_0},\frac{\mu}{\lambda}\right) = \frac{3}{4}$, $\varphi\left(\frac{3}{4}\right) \approx 0.03768$, 그리고 (13)은 0.29의 값을 가진다. $\mu = 1$로 바꾸고 나머지 매개변수들은 동일하다고 가정하면, $\max\left(\frac{\mu t_0}{w_0},\frac{\mu}{\lambda}\right) = \frac{3}{8}$, $\varphi\left(\frac{3}{8}\right) \approx 0.3558$, 그리고 (13)은 약 0.00001135의 아주 다른 값을 가지게 된다.

위 논의에 따르면, 부등식 $\lambda > \mu$이 성립하여야지만 시스템이 안전하다. 다른 말로, 정직한 거래들의 유입량이 공격자의 전산능력에 비해 커야 한다. 아니면, 근사치 (13)은 쓸모없게 된다. 이것은 탱글 기반 시스템의 초창기에는 체크포인트(checkpoint)같은 추가적인 보안 조치들이 필요하다는 것을 알려준다.

충돌하는 거래 두 개 중 어느 것이 유효한지 결정하는 방법을 선택할 때, 누적 무게 값을 통해 결정하는 것에 대해 조심해야 한다. 이것은 누적 무게는 섹션 4.1에 나온 공격과 비슷한 공격, 특히 공격자가 미리 이중지불 거래를 준비한 뒤, 비밀리에 그 거래를 참고하는 부분탱글을 만들고, 상인이 진짜 거래를 받아들인 뒤에 부분탱글을 알리는(broadcast) 방법에 취약할 수 있기 때문이다. 두 충돌하는 거래 중 하나를 선택하는 더 좋은 방법은 다음 섹션에 설명된 방법일 수도 있다: 끝점 선택 알고리즘을 돌린 뒤 두 거래들 중 어느 거래가 선택된 끝점에 의해 간접적으로 승인되는지 보는 것이다.

4.1 기생 체인을 통한 공격과 새로운 끝점 선택 알고리즘

(A Parasite Chain Attack and a New Tip Selection Algorithm)

다음과 같은 공격을 생각해보자 (그림 6): 어떤 공격자가 더 높은 점수를 얻기 위해 때때로 주 탱글을 참고하는 부분탱글을 비밀리에 만든다. 여기서 정직한 끝점들의 점수는 대략 주 탱글의 모든 자체 무게들의 합이고, 공격자의 끝점들의 점수 또한 대략 기생 체인의 모든 자체 무게들의 합이다. 혼자서 부분탱글을 만드는 공격자에게 네트워크 지연은 문제가 되지 않기에[3 2], 만약 공격자들이 충분히 강력한 컴퓨터를 사용한다면 기생 끝점들에게 값이 더 큰 높이를 줄 수도 있을 것이다. 또한, 공격자는 기생 체인에 더 일찍 요청되었던 거래들을 승인하는 새 거래들을 많이 요청함으로써 공격의 순간에 자신의 끝점 개수를 인위적으로 늘릴 수 있다 (그림 6). 이것은 정직한

[3 2] 이것은 공격자는 나머지 네트워크로부터의 정보에 의존하지 않고 항상 자신의 거래들을 승인할 수 있기 때문이다.

노드들이 간단한 선택과정을 통해 끝점을 고를 경우에 공격자에게 이점이 될 것이다.

이러한 형식의 공격으로부터 방어하기 위해서, 주 탱글이 공격자보다 더 많은 해싱 능력을 가져야 한다는 사실을 이용할 것이다. 따라서, 주 탱글은 공격자보다 더 많은 거래들의 누적 무게를 더 많이 증가시킬 수 있다. 여기서 승인할 두 끝점을 고르기 위해 MCMC 알고리즘을 사용하는 것이 아이디어이다.

Site의 현재 누적 무게를 H_x라고 하고, 모든 자체 무게는 1이라고 가정하였다는 것을 회상하자. 따라서, 끝점의 누적 무게는 항상 1이고, 다른 site들의 누적 무게는 적어도 2이다.

여기서 아이디어는 탱글의 site들에 몇 개의 입자(particle)들, 또는 무작위 워커들을 놓고, 무작위적인[33] 경로로 끝점들을 향해 행보하게 하는 것이다. 행보를 통해 선택된 끝점들은 승인될 후보가 된다. 이 알고리즘은 다음과 같다:

1. W가 충분히 클 때, 구간 $[W, 2W]$의 모든 site들을 고려한다[34]
2. 그 구간에 N개의 입자들을 독립적으로 놓는다[35].
3. 그 입자들이 끝점들을 향해 서로 독립적인 이산 시간 무작위 행보(discrete-time random walk)를 행하게 한다, 이는 y가 x를 승인할 때에만 x에서 y로의 전이(transition)가 가능하다는 것을 뜻한다.
4. 가장 먼저 끝점 집합에 도달한 두개의 무작위 워커들은 승인될 두 끝점들에 있을 것이다. 하지만, 이 규칙을 다음과 같이 변화시키는 것이 나을 수도 있다: 먼저 끝점들에 너무 빨리 도달한 무작위 워커들은 게으른 끝점들 중 하나에 도달했을 수도 있기 때문에 버린다.
5. 워커들의 전이 확률은 다음과 같이 정의된다: 만약 y가 x를 승인하면 $(\mathrm{y} \to \mathrm{x})$, 전이 확률 P_{xy}는 $\exp\left(-\alpha\left(H_x - H_y\right)\right)$에 비례하고, 다음 값을 가진다.

$$P_{xy} = \exp\left(-\alpha\left(H_x - H_y\right)\right)\left(\sum_{z:z\to x} \exp\left(-\alpha(H_x - H_z)\right)\right)^{-1} \tag{14}$$

---

[33] 무작위성의 정식 원천은 존재하지 않는다. 노드들은 자신의 (슈도)랜덤 숫자 생성기를 사용하여 무작위 워크를 행한다.

[34] 입자를 탱글 깊숙이 놓아 곧바로 끝점에 도달하지 않도록 하는 것이 핵심이다. 하지만, 적당한 시간 내에 끝점을 찾아야 하므로, 입자는 너무 깊숙이 놓여서는 안된다. 또한, 구간 $[W, 2W]$은 임의적으로 정해진 것으로, $[W, 5W]$등 또한 사용될 수 있다. 워커의 시작 지점을 고르는 다른 방법들도 존재한다. 예를 들어, t_0가 어떤 고정된 시각일 때, 노드는 단순히 과거 시간 t_0와 $2t_0$ 사이의 거래 하나를 무작위로 골라 시작지점으로 사용할 수 있다.

[35] 이 선택은 대체로 임의적이다. 우리는 추가적 보안을 위해 단 두 개의 입자 대신 여러 개의 입자들을 사용한다. 입자가 우연히 공격자의 체인으로 이동하게 된다면, 공격자의 체인이 더 길기 때문에 입자는 그곳에서 오랜 시간을 보내게 되고, 다른 끝점들이 먼저 선택되게 되는 것이 핵심이다.

이때 $\alpha > 0$는 선택되어야 하는 매개변수이다[36].

이 알고리즘은 지역적이어서, 연관된 계산들을 하기 위해 제네시스 거래까지 탱글을 되돌아가지 않아도 된다. 특히, 탱글 전체에 대해 누적 무게를 계산하지 않아도 된다. 최대로 해야하는 일은 단지 워커의 시작점을 간접적으로 승인하는 site들의 누적 무게를 계산하는 것에 불과하다.

알고리즘이 의도한대로 작동하는지 확인하기 위해서, 먼저 게으른 끝점들에 대해 생각해보자. 이 끝점들은 검증작업을 하지 않기 위해 일부로 오래된 거래들을 승인한다 (그림 6). 입자가 게으른 끝점에 의해 승인된 site에 있어도, 누적 무게의 차이가 아주 커, P_{xy}가 작을 것이기 때문에 게으른 끝점이 선택될 확률은 낮다..

다음으로, 이런 공격 방식을 고려해보자: 공격자가 비밀리에 자신의 지배 하에 그림 6의 좌측 붉은 원으로 나타내진 것처럼 자신의 계좌잔액을 다른 계좌로 비우는 거래를 포함하는 체인을 만든다. 그 다음, 공격자는 우측의 붉은 원으로 나타내진 것과 같이 주 탱글에 거래를 요청하고, 상인이 그것을 받아들이기를 기다린다. 여기서 기생 체인은 때때로 주 탱글을 참고한다. 하지만, 기생 체인에서의 누적 무게는 아주 크진 않고, 기생 체인은 상인에게의 거래 이후 주 탱글을 참고할 수 없다. 또한, 공격자는 공격을 하는 순간 인위적으로 자신의 기생 체인에 있는 끝점들의 개수를 늘리려고 시도할 수 있다 (그림 6). 공격자의 목표는 신규 거래를 요청하는 노드들이 기생 체인을 참고함으로써 탱글의 정직한 가지가 버려지게 하는 것이다.

MCMC 알고리즘이 왜 높은 확률로 공격자의 끝점을 선택하지 않을 것인지 쉽게 알 수 있다. 그 이유는 게으른 끝점 시나리오때와 비슷하다: 기생 체인의 site들은 그들이 참고하는 주 탱글의 site들보다 훨씬 작은 누적 무게를 가질 것이다. 따라서, 무작위 워커는 시작을 기생 체인에서 하지 않는 이상, 기생 체인으로 이동할 확률은 거의 없고, 주 탱글이 더 많은 site들을 가지고 있기 때문에 기생 체인에서 시작할 확률 또한 거의 없다.

왜 노드들이 이 알고리즘을 따를 것인지 논평해보자. 섹션 1에서 상당수의 노드들은 어떤 참조 알고리즘을 따를 것이라는 했던 것을 기억하자. 또한, 전산적 지연, 네트워크 지연으로 인해, 끝점 선택 알고리즘은 거래의 요청되는 시점 이전의 탱글 스냅샷을 사용하게 될 것이다. 뒤에 설명할 이유들로 인해, 참조 알고리즘에서 일부러 더 과거의 스냅샷을[37] 사용하는 것이 나을 수도 있다. 자신의 거래가 빨리 승인될 확률을 최대화 시키려고만 하는 이기적인 노드를 생각해보자. 노드들의 상당수에 의해 사용되는 이번 섹션의 MCMC 알고리즘은 끝점 집합에 승인 확률 분포를 준다. 이기적인 노드는 자연적으로 그 분포의 최대값들을 가지는 끝점들을 선택할 것이다. 하지만, 만약

[36] 이것은 $\alpha = 1$에서부터 시작하여도 된다.

[37] 먼저 무작위 워커는 그 스냅샷에서 과거의 끝점을 찾고 현재 탱글에서 "현재의" 끝점들을 향해 행보한다.

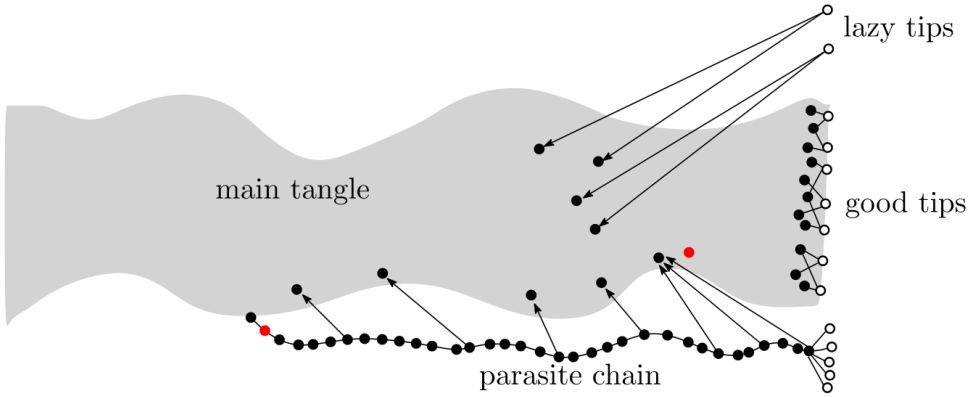

그림 6: 정직한 끝점들과 기생 체인의 끝점 선택 알고리즘의 시각적 표현. 두 붉은 원은 공격자의 이중지불 시도를 나타낸다.

다른 많은 노드들 또한 이기적이게 행동하고 동일한 방법을 사용하는 충분히 합리적인 상황이 된다면, 그들은 모두 실패할 것이다. 많은 신규 거래들은 거의 같은 시각에 동일한 두 끝점을 승인할 것이고, 따라서 뒤따르는 승인에 대해서 그들간에 너무 많은 경쟁을 만들 것이다. 노드들은 과거의 스냅샷을 사용하기 때문에 동일한 두 끝점의 대량 승인으로 인한 누적 무게의 증가를 즉각적으로 알지 못할 것이다. 이 이유 때문에, 이기적인 노드라도 참조 끝점 선택 알고리즘에 의해 생성된 디폴트(default) 확률 분포와 비슷한 끝점 선택 확률분포를 가지는 어떤 무작위 끝점 선택 알고리즘을[38] 사용해야 할 것이다. 이기적인 노드의 존재 하에 이 "모여있는(aggregated)" 확률분포가 디폴트 확률분포와 동일할 것이라고 말하는 것이 아니다. 다만, 위 주장은 두 확률분포가 아주 비슷해야 한다는 것을 보여준다. 이것은 많은 노드들이 동일한 나쁜 질의 끝점들을 검증하는 것을 시도할 확률은 작을 것을 뜻한다. 어떤 경우에도, 가능한 이익은 컨퍼메이션 시간이 살짝 감소하는 것에 불과하기 때문에 노드들은 이기적인 것에 대한 큰 인센티브가 없다. 이것은 비트코인과 같은 다른 분산화된 구조(construct)와 본질적으로 다르다. 중요한 사실은 바로 노드들은 MCMC 끝점 선택 알고리즘을 버릴 이유가 없다는 것이다.

(14)에 주어진 것과 같이 전이 확률의 정의는 확고하게 정해진 것이 아니다. 지수함수 대신, $f(s) = s^{-3}$과 같이 빠르게 감소하는 다른 함수를 사용할 수도 있다. W와 N또한 자유롭게 선택될 수 있다. 현재 이 매개변수들이 정확히 어떻게 선택되어야 하는지 보여주는 어떤 이론적 주장이

[38] MCMC를 여러 번 반복하는 방법 외에 어떤 끝점들이 더 좋은지 (정직한 노드들에 의해 선택될 확률이 높은지) 쉽게 찾을 수 있는 방법은 없어 보인다. 하지만, MCRW를 여러 번 행하는 것은 시간과 다른 자원들을 필요로 한다; 노드가 이것에 시간을 좀 쓴 후, 탱글의 상태는 이미 바뀌기 때문에 노드는 아마 처음부터 다시 시작해야 할 것이다. 이것은 다른 노드들의 상당수가 디폴트 끝점 선택 방법을 사용할 때, 왜 노드들이 다른 방법을 쓰기 위해 MCMC 끝점 선택 방법을 버릴 이유가 없다는 것을 설명해준다.

존재하는지는 확실하지 않다. 결론적으로, 이 섹션의 주된 기여는 MCMC를 끝점 선택에 사용한다는 아이디어라고 생각된다.

4.2 분리 공격 (Splitting attack)

제안된 MCMC 알고리즘에 대한 다음 공격은 Avic Zohar에 의해 제안되었다. 고부하 상태일 때, 공격자는 탱글을 두 개의 가지(branch)로 분리하고 둘 사이의 균형을 유지하려고 할 수 있다. 이것은 두 개의 가지 모두 계속 자랄 수 있게 할 것이다. 정직한 노드가 동시에 두 가지를 참고하여 두 가지를 잇는 것을 방지하기 위해, 공격자는 분리의 시작지점에 적어도 두 개의 충돌하는 거래를 놓아야 한다. 그런 다음, 비교적 적은 개인적 전산능력으로도 무작위적 변동(random fluctuations)을 보정(compensate) 할 수 있도록 공격자는 각 가지마다 네트워크의 절반 정도가 일을 하기를 기대한다. 이 방법이 성공한다면, 공격자는 같은 돈을 두 개의 가지에서 사용할 수 있게 된다.

이러한 공격으로부터 방어하기 위해서는, 두 가지의 균형을 유지하는 것을 너무 힘들게 하는 "가파른 임계치(sharp threshold)" 규칙이 있어야 한다. 이런 규칙의 예 중 하나는 비트코인 네트워크에서 가장 긴 체인을 선택하는 규칙이다. 이 개념을 분리 공격 하에 있는 탱글에서 해석해보자. 먼저 첫 번째 가지의 총 무게는 537이고, 두 번째 가지의 총 무게는 528이라고 가정하자. 정직한 노드가 $1/2$에 아주 가까운 확률로 첫 번째 가지를 선택한다고 하면, 공격자는 아마 두 가지간의 균형을 유지할 수 있을 것이다. 하지만, 만약 정직한 노드가 첫 번째 가지를 $1/2$보다 훨씬 큰 확률로 선택한다면, 공격자는 아마 균형을 유지하지 못할 것이다. 후자의 경우에서 두 가지 간의 균형을 유지하지 못하는 이유는 필연적으로 일어나게 될 무작위 변동 이후에, 네트워크는 빠르게 한 개의 가지를 고르고 다른 하나는 버리게 될 것이기 때문이다. MCMC 알고리즘이 이렇게 행동하게 만들기 위해서는, 아주 빠르게 감쇠하는 함수 f를 고르고, 행보가 가지의 분기점 이전에 시작할 가능성이 크도록 깊이 값이 큰 노드에서 무작위 행보를 시작해야 한다. 이런 경우에, 무작위 행보는 경쟁하는 가지 간 누적 무게값의 차이가 작아도 높은 확률로 "더 무거운" 가지를 선택할 것이다.

네트워크 동기화 문제 때문에도 공격이 매우 어렵다: 공격자는 많은 수의 최근에 요청된 거래들에 대해 모를 수도 있다[39]. 분리 공격으로부터 방어하는 또 다른 효과적인 방법은 충분히 강력한 자가 많은 수의 거래들을 하나의 가지에 한순간에 요청하여, 가지 간 균형을 급격히 변화시켜 공격자가 이 변화에 대처하기 어렵게 하는 것이다. 만약 공격자가 분리를 유지시켜도, 가장 최근

[39] 실제 누적 무게는 그들이 생각하는 값과 꽤 다를 수도 있다.

의 거래들은 겨우 50% 정도밖에 되지않는 컨퍼메이션 확신도를 가질 것이고 (섹션 1), 가지들은 성장하지 않을 것이다. 이 경우에, 정직한 노드들은 분기점 이전의 거래들만 선택적으로 승인하여 나뉜 가지들에 있는 충돌하는 거래들의 승인을 우회할 수 있다.

끝점 선택 알고리즘의 다른 버전들을 고려할 수도 있다. 예를 들어, 만약 노드가 두 개의 큰 부분탱글과 마주치게 된다면, 먼저 자체 무게들의 합이 더 큰 부분탱글을 선택하게 한 뒤, 위의 MCMC 끝점 선택 알고리즘을 이행할 수도 있다.

훗날에서의 사용을 위해 다음 아이디어도 고려할 가치가 있을 수도 있다. (14)에서 정의되어 있는 전이 확률을 워커가 탱글 깊숙이 있으면 마르코프 체인의 다음 단계가 거의 예측적이게, 워커가 끝점들에 가까이 있으면 더 무작위적이도록 $H_x - H_y$와 H_x 두가지 모두에 의존하게 만들 수 있다. 이것은 승인할 두 끝점을 고를 때 충분한 무작위성을 가지면서도 더 약한 가지를 선택하지 않도록 도울 것이다.

결론들:

1. 공격자가 시스템을 앞지름으로써 이중지불을 시도하는 공격 방법들을 몇가지를 고려해보았다.

2. "큰 무게" 공격은 이중지불을 하기 위해서, 공격자가 이중지불 거래에 아주 큰 무게를 부여하는 것을 시도하여 그것이 진짜 탱글보다 무겁게 하는 것이다. 이 방법은 허용되는 자체 무게에 제한이 없을 경우에 네트워크에 위협이 된다. 해결책으로는, 거래의 자체 무게에 제한을 두거나, 상수값으로 고정할 수 있다.

3. 거래가 가질 수 있는 최대 자체 무게값이 m인 상황에서, 최선의 공격 방법은 이중지불 거래를 참고하는 자체 무게 m의 거래들을 생성시키는 것이다. 공격자의 전산능력에 비해 유입되는 정직한 거래들이 충분히 많으면, 공식 (13)을 통해 이중지불 거래가 더 큰 누적 무게를 가질 확률을 예측 할 수 있다 ((13) 아래의 예제 또한 보아라).

4. "기생 체인"을 통한 공격은 공격자의 site들이 진짜 탱글에 비해 높이와 점수와 같은 단위가 높은 값들을 갖기 때문에 높이나 점수에 기반한 승인 방법을 쓸모없게 한다. 반면에, 섹션 4.1에서 설명한 MCMC 끝점 선택 알고리즘은 이러한 공격으로부터 보호해줄 것으로 보인다.

5. MCMC 끝점 선택 알고리즘은 보너스로 네트워크를 게으른 노드들로부터 보호해주기도 한다.

5. 양자컴퓨터에 대한 저항성 (Resistance to Quantum Computations)

충분히 강력한 양자컴퓨터(quantum computer)는[40] 반복적인 시도에 의존하여 답을 찾는 문제의 해결에 아주 효율적일 수 있다고 알려져 있다. 이러한 반복적인 시도에 의존하는 문제의 좋은 예는 바로 비트코인 블록을 생성하기 위해 넌스를 찾는 과정이다. 현재 새로운 블록의 형성을 가능하게 하는 적당한 해시값을 찾기 위해서는 평균적으로 2^{68}개의 넌스들을 확인해야 한다. 양자컴퓨터는 위의 비트코인 퍼즐과 유사한 문제를 풀기 위해서는 $\Theta(\sqrt{N})$번의 연산(operation)이 필요하다고 알려져 있다 (예를 위해서는 [15]를 보라). 반면에 고전적인 컴퓨터에서 동일한 문제를 풀기 위해서는 $\Theta(N)$번의 연산을 필요로 한다. 따라서, 양자컴퓨터는 고전적인 컴퓨터보다 $\sqrt{2^{68}} = 2^{34} \approx$ 170억배 정도 비트코인 채굴에 더 효율적일 것이라고 할 수 있다. 또한, 만약 블록체인이 증가한 해싱 능력(hashing power)에 따라 난이도를 증가시키지 않는다면 버려지는 블록들의 rate가 증가할 것이다.

같은 이유로, "큰 무게" 공격은 양자컴퓨터를 사용하면 훨씬 더 효율적일 것이다. 하지만, 섹션 4에서 제안한 것처럼 무게에 제한을 두면, 양자컴퓨터를 통한 공격 또한 효과적으로 방어할 수 있을 것이다. Iota에서는 거래를 요청하기 위한 알맞은 해시값을 찾기 위해 확인하여 할 넌스들의 개수가 많지 않기 때문에 이 공격에 대한 방어능력은 iota에서 더욱 분명하게 나타난다. iota에서 일반적으로 확인하여야 할 넌스의 개수는 3^8개 정도다. 따라서 이상적인 양자컴퓨터가 얻는 효율의 증가는 $3^4 = 81$배 정도로, 수용 가능한 정도이다[41]. 더 나아가, iota에 적용된 알고리즘은 넌스를 찾는데 걸리는 시간이 거래를 요청하기 위해 해야 하는 다른 일들에 걸리는 시간보다 별로 길지 않은 구조를 가진다. 후자는 양자컴퓨터에 대해 훨씬 더 큰 저항성을 가지며, 따라서 (비트코인) 블록체인과 비교하였을 때, 양자컴퓨터를 가진 적(adversary)으로부터 훨씬 더 뛰어난 보안을 가진다.

감사의 글 (Acknowledgements)

필자는 초안에서 몇가지 오류들을 지적해준 Cyril Grünspan과 Toru Kazama, 그리고 이 백서를 더 읽기 쉽게 만드는 것을 도와준 James Brogan에게 감사의 뜻을 전합니다.

[40] 오늘날로써 아직도 이론상에만 존재한다.

[41] $\Theta(\sqrt{N})$은 쉽게 $10\sqrt{N}$의 값을 가질 수 있다는 것을 알자.

참고문헌 (References)

[1] Iota: a cryptocurrency for Internet-of-Things. See **http://www.iotatoken.com/,** and **https://bitcointalk.org/index.php?topic=1216479.0**

[2] bitcoinj. Working with micropayment channels. **https://bitcoinj.github.io/working-with-micropayments**

[3] people on nxtforum.org (2014) DAG, a generalized blockchain. **https://nxtforum.org/proof-of-stake-algorithm/dag-a-generalized-blockchain/** (registration at **nxtforum.org** required)

[4] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, Aviv Zohar (2012) On Bitcoin and red balloons. *Proc. 13th ACM Conf. Electronic Commerce*, 56-73.

[5] Richard Durrett (2004) Probability - Theory and Examples. *Duxbury advanced series.*

[6] Sergio Demian Lerner (2015) DagCoin: a cryptocurrency without blocks. **https://bitslog.wordpress.com/2015/09/11/dagcoin/**

[7] Yonatan Sompolinsky, Aviv Zohar (2013) Accelerating Bitcoin's Transaction Processing. Fast Money Grows on Trees, Not Chains. **https://eprint.iacr.org/2013/881.pdf**

[8] Yonatan Sompolinsky, Yoad Lewenberg, Aviv Zohar (2016) SPECTRE: Serialization of Proof-of-work Events: Confirming Transactions via Recursive Elections. **https://eprint.iacr.org/2016/1159.pdf**

[9] Yoad Lewenberg, Yonatan Sompolinsky, Aviv Zohar (2015) Inclusive Block Chain Protocols. **http://www.cs.huji.ac.il/~avivz/pubs/15/inclusive btc.pdf**

[10] Joseph Poon, Thaddeus Dryja (2016) The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. **https://lightning.network/lightning-network-paper.pdf**

[11] Sheldon M. Ross (2012) *Introduction to Probability Models*. 10th ed.

[12] David Vorick (2015) Getting rid of blocks. **slides.com/davidvorick/braids**

[13] Amir Dembo, Ofer Zeitouni (2010) *Large Deviations Techniques and Applications*. Springer.

[14] Sheldon M. Ross (2009) *A First Course in Probability*. 8th ed.

[15] Gilles Brassard, Peter Hyer, Alain Tapp (1998) Quantum cryptanalysis of hash and claw-free functions. *Lecture Notes in Computer Science* **1380**, 163-169.

수정내역

날짜: 성함, 수정내용, 주소, 시그니처(날짜, 성함, 수정내용)

1. 2017/09/20: ChanHyun Park (Caffe1ne4Life), IOTA 백서 v1.2 한국어 번역

   주소: 1L7NA876yhVwQnSJNQwkuXJHDyQLpsSZGu

   시그니처: IM2ArZLttV65kGzNn0x6raa4sV9Y73Eu+1f0UxrUeNv8JW4escxMqxw+mxzJbBpnLi48vxoJV/MbAW4B8gOuF3M=