ブロックチェーン公開講座 第4回

ビットコインその3

芝野恭平

東京大学大学院工学系研究科技術経営戦略学専攻
ブロックチェーンイノベーション寄付講座
特任研究員
shibano@tmi.t.u-tokyo.ac.jp

マイニングとコンセンサス

概要

- ブロックチェーンは改ざんできないことはわかったが，新しいブロックに含まれる情報がなぜ正しいものがはいるのか，その正しさはどのように全体でコンセンサスを得るのか，をここで説明する.

- 合意のプロセス
- トランザクション検証
- ブロック生成・マイニング
- ブロック検証
- ブロックチェーンの選択

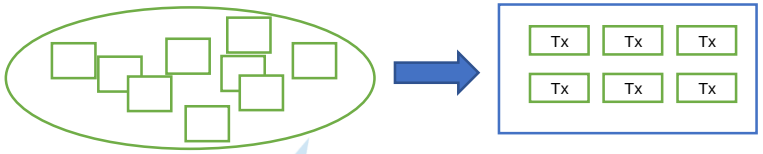
ブロックとPoW

ブロックは世界で1個ずつ生成されてほしい

- 一度に複数のブロックがどんどん提案されていくとどうなるか？
  - 世界中にノードが存在し，トランザクションプールの中身も異なる．
  - 複数のブロックが同時に作られると，矛盾がないように採用するブロックを調整していくのは難しそう．
- 世界で1個ずつブロックが順番に生成されるようにしたい．
  - こうすれば，新しくできたブロックを都度「正しい」かどうかを判断していけばよくなる．

生成されるブロック内には，正しいトランザクションのみが入ってほしい

- 正しいブロックを生成したときには報酬を，正しくないブロックを提案する人には罰則を設けたい．
  - ブロック生成を促すための報酬．
  - 不正なブロックが生成されてしまうことを防ぐための罰則．



PoWは上記2つの問題を解決している

- ブロックを提案するためには，計算を一定量注ぎ込む，ということを強制し，成功したら報酬を与える．
  - 計算をするためには電気代がかかり，それが一種罰則となる．
- この計算難易度をうまく調整し，定期的に世界で1個ずつブロックが生成されるようにする．
- 一定の計算量を注いだそのブロックに入っているトランザクションは「正しい」と胸を張って言えるものしか生成がされなくなる．
  - 正しくないブロックは他のノードで破棄されてしまうため，計算にかかった電気代が無駄になる．
- 正しいブロックが生成できるとマイニング報酬がもらえる．

分散化コンセンサス

- どのようにして中央管理者なしに「正しさ」に「合意」するのか
  - 中央集権型のシステムの場合はその運営者が正しさのチェックをする.
  - 例：クレジットカードの支払いではその運営会社が正しさを保証し，ユーザーは運営会社を信用する.
- 分散化されたコンセンサス
  - 個々のノードが独立して動いていても全体として一つのブロックチェーンが正しいとみなされる.
    - コンセンサスの創発とも呼ばれる.
    - ルールブック（検証しなければならない項目リスト）に沿って行動をすることを求める.
    - ルールブックに従わなかった場合は，他のノードで無効なものとして除外されてしまう.
- 以下の4つの相互作用で成り立っている.
  - 独立したトランザクション検証
    - すべてのフルノードは，受け取ったトランザクションについて，包括的なリストに基づく検証を行う
    - 要は，一つ一つのノードそれぞれで，共有されている一つの検証方法に沿って，検証できるようになっている
  - 独立したトランザクション集積
    - PoWアルゴリズムによる計算と結びついた，マイニングノードによるブロックへのトランザクションの集積
    - 要は，マイニングノードがブロックに入れるトランザクションリストを自由に決めてよい，ということ.
  - 独立した新規ブロック検証とブロックチェーンへの埋め込み
    - すべてのノードによって包括的なリストに基づくブロック検証がされ，そのブロックを自身が持つブロックチェーンに取り込む.
  - 独立したブロックチェーン選択
    - 個々のノードでPoWを通じて証明された，最も多くの累積計算量を持っているブロックチェーンが選ばれる.
    - 要は，フォークしているブロックチェーンがいくつかある場合にどれを選ぶかは，個々のノードごとに認識しているブロックチェーンの中で累積計算量を見て決めている，ということ.
    - これは最長のチェーンとなる.

独立したトランザクション検証

- 各ノードで，伝搬前にトランザクションが検証され，不正なトランザクションは伝搬されない．
- 条件は以下：

トランザクションの構文とデータ構造は正しいか
インプットとアウトプットのいずれも空でないか
トランザクションデータサイズがMAX_BLOCK_SIZEバイト未満か．（ブロックの最大サイズ）
それぞれのアウトプット値およびtotal valueは許容範囲内の値(0より大きく2,100万BTCより小さい)か
インプットのいずれもhash=0, N=-1でないか(coinbaseトランザクションでないか)
nLockTimeはINT_MAX以下か
トランザクションデータサイズは100Bytes以上か．
トランザクションに含まれている署名操作（SIGOPS）の数は回数上限よりも小さいか
アンロックスクリプトはスタックに数字をpushすることしかできず，ロッキングスクリプトはisStandard形式に合っているか． → アンロッキング，ロッキングスクリプトが「標準形式」か
トランザクションプールまたはブロックチェーンのブロックに同じトランザクションがないか．

各インプットに対して，もし参照しているアウトプットをトランザクションプールの他のトランザクションが参照していないか．
各インプットに対して，ブロックチェーンかトランザクションプールにインプットが参照しているアウトプットが見つかるか確認する．もし見つからなければオーファントランザクションであり，オーファントランザクションプールにこのトランザクションがない場合は追加する．
各インプットに対して，もしインプットが参照しているアウトプットがcoinbaseアウトプットだった場合は，COINBASE_MATURITY(100）の承認数を持っているか（100ブロック進んでいるか）
各インプットに対して，参照しているアウトプットが既に使用されていないか．
参照しているアウトプットを使って，それぞれのインプットvalueとその総和が許容範囲にあるか
インプットvalueの総和がアウトプットvalueの総和よりも大きいか
トランザクション手数料が少なすぎないか(minRelayTxFeeでクライアントごとに定められている．0.00001BTC or 0.1 BTC/KB)
各インプットのアンロッキングスクリプトは，対応したアウトプットのロッキングスクリプトを解除できるか．

マイニングノード：マイニングを行っているノード

- ノードのうちマイニングを行っているノードで，マイナーとも呼ばれる．
  - PoWの解を求める作業を行います．

- 必ずしもフルノードでなくともよい．
  - マイニングプールではフルノードでなくともマイニングする．
  - 多くの場合，フルノードでもありすべてブロックの検証が可能な状態のブロックチェーンに対してマイニングする．
  - 万が一マイニングに成功したものの，生成したブロックが他のノードでの検証に失敗すると，そのノードでブロックチェーンに取り込まれなくなってしまうし他のノードに伝搬もされない．

- マイニングノードはマイニングに成功すると，マイニング報酬とそのブロックに含まれているトランザクションの手数料すべてを受け取れる．

- 新規ブロックが他のノードから伝搬された場合，現在のマイニングに負けたことを意味するため，その次のブロックのマイニングを始める必要がある．
  - 最長ブロックのブロックチェーンが正しい，とみなされるため対抗しても負ける可能性が高い．
  - 他のマイナーは，すでにそのブロックからマイニングをしている可能性もある．

マイニング報酬と手数料

- マイニング成功時に得られる報酬は，マイニング報酬＋手数料

- マイニング報酬
  - マイニング報酬とはすなわちビットコインの新規発行分.
  - 210,000ブロックごとに報酬は半減する.
    - 約4年，1ブロック=10分
  - 最初50BTCで，2024年4月には3.125BTCになった.
    - 2024/04現在1BTC≒1,000万円なので約3,125万円.
  - 最終的にマイニング報酬が0になるのは2140年ごろ
    - 1 satoshiを切る
  - そのとき総発行量は2100万BTCとなる.
- 手数料
  - ブロック内に入れるトランザクション全体の手数料総和

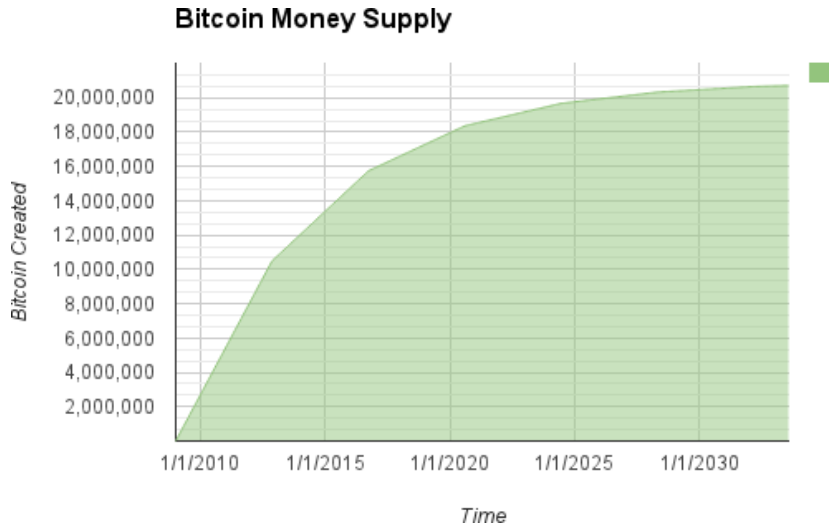
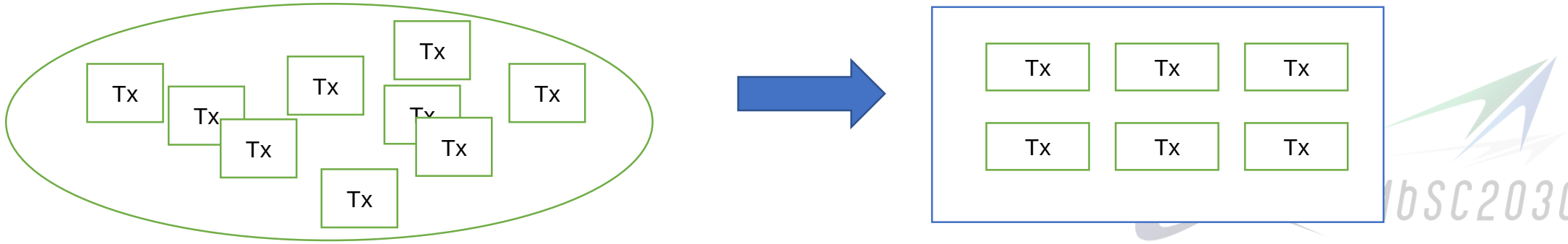


Figure 1. Supply of bitcoin currency over time based on a geometrically decreasing issuance rate

ブロックへのトランザクション集積

- マイニングノードは新しいブロックを作る
  - ブロックの構造はブロックヘッダー＋トランザクションのリスト
- トランザクションリストを作ることを先に考える．
  - ブロック内に，どのトランザクションを入れるかを選ぶ．
  - トランザクションは自身のトランザクションプールから選ぶ．
- マイナーは自分自身の報酬を最大化する．
  - 容量単位の手数料がよいトランザクションから取り込む．
- トランザクションプールからブロックに含めるトランザクションを選ぶのに加え，Coinbaseトランザクションを作る必要もある．



Coinbaseトランザクション

- ブロック内の最初のトランザクションをcoinbaseトランザクションもしくはgenerationトランザクションと呼ぶ.
- マイナー自身への支払いとなる，マイニング報酬＋手数料が含まれるトランザクション.
- インプットはcoinbaseと呼ばれるデータ領域になっている
  - マイニング報酬は無から生み出される，すなわち対応するUTXOはない
  - エクストラナンスとしても使用される（後述）
- アウトプットはマイナーのアドレスへ（公開鍵を含むロッキングスクリプト）.

Coinbaseトランザクションにおける Inputの構造 Inputトランザクション領域にUnlockingスクリプトは不要

```
$ bitcoin-cli decoderawtransaction
0200000000010100000000000000000000000000000000000000000000000000000
000000000ffffffff1f0318e318045396fc5d434e2f544553544e455464000000e36a020000
000000ffffffff02e40b54020000000017a914c471255f1e96484541cef0b3d6b299fc63daa
1c8870000000000000000266a24aa21a9ede2f61c3f71d1defd3fa999dfa36953755c690
689799962b48bebd836974e8cf90120000000000000000000000000000000000000000
00000000000000000000000000000000
```


```
{
  "txid": "e3227e2b306d45d2f92d472ad50abc4b6e11e1f823f4ca9fb97bb37dc32f7d0a",
  "hash": "6e93afd1d61f1034810f302bdb6a91d2016a9273965d283e0a5f89658099f386",
  "version": 2,
  "size": 197,
  "vsize": 170,
  "locktime": 0,
  "vin": [
    {
      "coinbase": "0318e318045396fc5d434e2f544553544e455464000000e36a020000000000",
      "sequence": 4294967295
    }
  ],
  "vout": [
    {
      "value": 0.39062500,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_HASH160 c471255f1e96484541cef0b3d6b299fc63daa1c8 OP_EQUAL",
        "hex": "a914c471255f1e96484541cef0b3d6b299fc63daa1c887",
        "reqSigs": 1,
        "type": "scripthash",
        "addresses": [
          "2NB9uyQML4CrC5pZ1ARqZ99n2P9twx8RqJg"
        ]
      }
    },
    {
      "value": 0.00000000,
      "n": 1,
      "scriptPubKey": {
        "asm": "OP_RETURN
aa21a9ede2f61c3f71d1defd3fa999dfa36953755c690689799962b48bebd836974e8cf9",
        "hex": "6a24aa21a9ede2f61c3f71d1defd3fa999dfa36953755c690689799962b48bebd836974e8cf9",
        "type": "nulldata"
      }
    }
  ]
}
```


トランザクションのコインベースInputの構造は以下.
任意のデータを入れられる領域が存在.

Size	Field	Description
32 bytes	Transaction Hash	All bits are zero: Not a transaction hash reference
4 bytes	Output Index	All bits are ones: 0xFFFFFFFF
1–9 bytes (VarInt)	Coinbase Data Size	Length of the coinbase data, from 2 to 100 bytes
Variable	Coinbase Data	Arbitrary data used for extra nonce and mining tags. In v2 blocks; must begin with block height
4 bytes	Sequence Number	Set to 0xFFFFFFFF

Table 2. The structure of a coinbase transaction input

（参考）以下，通常のトランザクションインプット．構造は同じ．

Size	Field	Description
32 bytes	Transaction Hash	Pointer to the transaction containing the UTXO to be spent
4 bytes	Output Index	The index number of the UTXO to be spent, first one is 0
1–9 bytes (VarInt)	Unlocking-Script Size	Unlocking-Script length in bytes, to follow
Variable	Unlocking-Script	A script that fulfills the conditions of the UTXO locking script
4 bytes	Sequence Number	Usually set to 0xFFFFFFFF to opt out of BIP 125 and BIP 68

Table 1. The structure of a "normal" transaction input

Coinbaseトランザクションの構造
Inputトランザクション領域には自由にデータを入れられる

- Coinbaseトランザクションのインプットはアンロッキングスクリプトの代わりに任意のデータを入れられる.
  - Genesisブロックの新聞見出し
  - 2バイト～100バイト

- 最初にブロック高を入れ，そのあとは任意のデータ.
  - エキストラナンスとしても使用される.

ブロックヘッダの構築
ナンス以外の値は正しい値をいれておき，ナンスをPoWで求める

- トランザクションリストが決まったら，今後はブロックヘッダを構築する．
  - 正しいブロックヘッダを作る必要がある．
  - 他のノードで無効なブロックと認識されてしまうと，せっかくブロック生成に成功しても報酬がもらえなくなる．
- Version, Previous Block Hash, Merkle Root, Timestamp
  - 説明省略
- Target
  - PoWのdifficultyターゲット
- Nonce
  - マイニングで求める値．Difficultyターゲットによる値の範囲を満たすブロックハッシュとなる値を求める．
  - ここをPoWで求める．
  - 逆にここ以外は正しい値をあらかじめ入れて準備しておく．
    - エキストラナンス領域はNonceと同様

Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block’s transactions
4 bytes	Timestamp	The approximate creation time of this block (seconds from Unix Epoch)
4 bytes	Target	The Proof-of-Work algorithm target for this block
4 bytes	Nonce	A counter used for the Proof-of-Work algorithm

Table 3. The structure of the block header

difficultyターゲットとマイニング

- マイニングでは，条件を満たすナンス値(4 bytes)を発見する
- ブロックハッシュの値がターゲット未満になるようなナンス値を探す
  - ブロックヘッダのダブルハッシュ値(SHA256 2回)

  - 例えば，
  - 0x00000000000000003A30C00000000000000000000000000000000000000000
  - というターゲットの場合，ブロックハッシュの値がこの値より小さい必要がある．

difficultyターゲットの表現

- Difficultyターゲットはヘッダに含まれる
  - 4 Bytes
  - 初めの1バイトが指数(exponent)、残りの3バイトが係数(coefficient)

$$target = coefficient * 2^{(8*(exponent-3))}$$

- 0x1903a30cなら

  $target = 0x03a30c * 2^{0x08*(0x19-0x03)}$

  $target = 0x03a30c * 2^{(0x08*0x16)}$

  $target = 238,348 * 2^{8*22}$

  $target = 238,348 * 2^{176}$

  $target = 22,829,202,948,393,929,850,749,706,076,701,368,331,072,452,018,388,575,715,328$

  $target = 0x00000000000003A30C0000000000000000000000000000000000000000$

- ターゲットは小さければ小さいほど難易度は上がる

difficultyターゲットの更新

- ブロック生成時間が10分に保たれるようにターゲットが一定間隔で調整される.

- 「10分」という間隔は，何十年にもわたって保ちたい.
  - コンピュータの急激な進化にも耐えられるようにする.
  - マイニング参加者，コンピュータ台数も変化する

- ターゲットは2,016ブロックごと（約２週間ごと）に調整される.

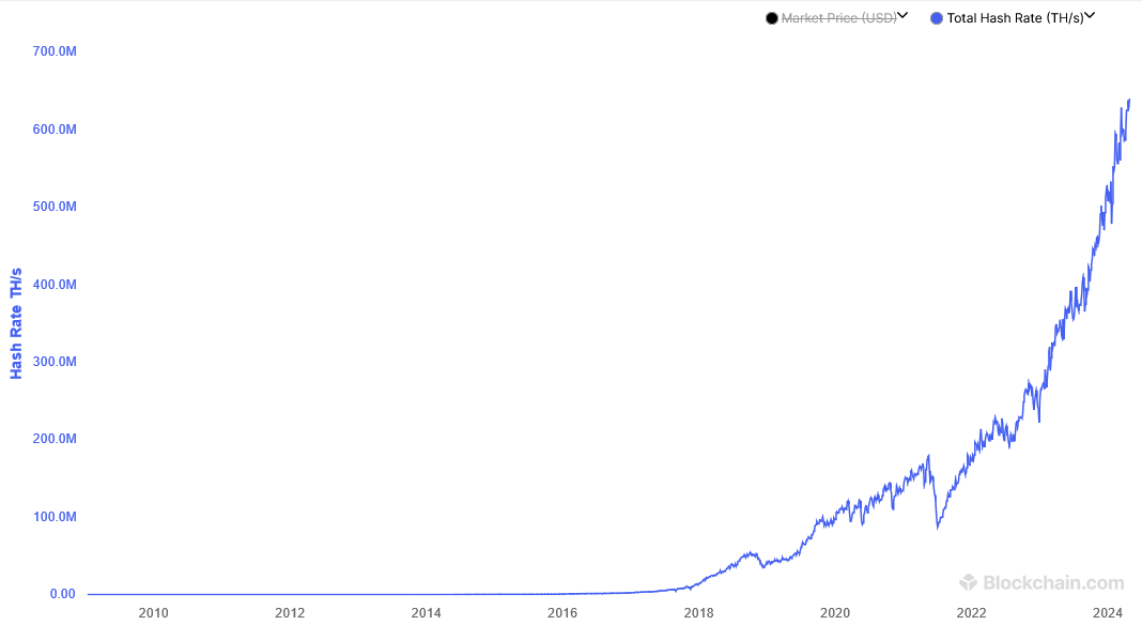
New Target = Old Target * (Actual Time of Last 2016 Blocks / 20160 minutes)

  - 例：過去の2,016ブロックでかかった時間が20,160分より短いとき
    - difficultyが簡単すぎ
    - 新しいターゲットは小さくなるように調整される.
    - より解が見つかりにくくなるように.

- ターゲットが極端に動きすぎないように，調整の際，最大4倍，最小で1/4以内になるようになっている.

マイニングとハッシュ化競争

- CPU → GPU/FPGA (2011年) → ASIC (2013年) と特化ハードウェアへ移行してきた.
  - CPU： 普通のパソコンに搭載
  - GPU： GPGPUで高速な並列計算が可能
  - FPGA: 回路をプログラム可能
  - ASIC：専用回路.
- ハッシュレート
  - 毎秒総生成ハッシュ数
  - ビットコインネットワーク全体のハッシュレートは以下：



https://www.blockchain.com/charts/hash-rate

- ハードウェアの進化とマイニングノードの増加により，指数関数的に増加している.
- これに伴いdifficultyも増加するので，新規参入は困難
  - 手元のPCで気軽にマイニングしても難しい.

マイニングのプログラム例

- トランザクションリストを選び，ブロックヘッダのナンス以外の部分を構築したら，候補ブロックに対してやることは，あとはマイニングのみ．

- ナンス値を変えて，ブロックハッシュを求めてターゲット以下になるか，ということを繰り返す．
  - ハッシュ関数は不可逆で，元の値を少しでも変えるとハッシュ値は大幅に変わる．
- Pythonの実装例：

```
def proof_of_work(difficulty):
    # nonceは4byte
    for nonce in range(256 ^ 4):
        header = get_header(nonce)
        blockhash = double_hash(header)
        if blockhash < difficulty:
            return nonce
    raise NotFoundException()
```

ナンス値について

- ナンス値の領域が4バイトって少なすぎない？
  - 4バイト正整数の最大値は約43億
- 4Bytesでは見つからないことがあるので，ブロックヘッダに含まれるほかの要素を変更していく．
- Coinbaseの文字列
  - マイニング報酬のトランザクションのvinにはcoinbaseが入る．
  - ここのデータ領域は最大100バイトの任意のデータを入れられる．
  - Extra nonceとも呼ばれる．
- Timestamp
  - Timestampは正確な時間を必ずしも入れる必要はなく，数秒，数十秒ずれたところで問題は生じない．
- ブロックに含むトランザクション
  - 取り込むトランザクションを変えてもよい．
  - 順番を変えるだけでもマークルルートの値が変わる．

2024-04-29のBlock #841,346を見てみるとブロックハッシュは
00000000000000000002a496fa0cf17d5c8f25b345ff9ad589b68381b02faa29
https://blockstream.info/block/00000000000000000002a496fa0cf17d5c8f25b345ff9ad589b68381b02faa29?expand

・この例だと，先頭9バイトが0x00
・仮にこの先頭9バイトが0x00となる元のインプットを求める問題とすると，ハッシュ値の解の候補は256^(32-9)
・出力ハッシュ値一個につき，解の候補に該当する確率は
256^(32-9) /(256^32)
= 1 / 4,722,366,482,869,645,213,696
・ 47垓分の1以下の確率（万，億，兆，京，垓）
・入力値で変更できる値が43億通りしかないと足らない．

マイニングに成功したら

- 正しい新しいブロックが生成できたということになる.
- これは，いち早く他のノードに伝搬すべき.
  - 他のマイナーも同時期にマイニングに成功している可能性があり，そちらが採用されてしまうと自分のマイニングしたブロックが無効になってしまうため.
  - そうすると，もらえるはずだった報酬がもらえなくなってしまう.

- 他のノードはどのように新しいブロックの検証をやっているのか？

新しいブロックの検証

- 各ノードは受け取った新規ブロックについて，他のノードへ伝搬する前に独立に検証を行う．

ブロックのデータ構造が正しいこと
ブロックハッシュがターゲットより小さいこと
ブロックのタイムスタンプが，ノードが持つ時間より2時間未来の時間よりも小さいこと
ブロックサイズが受け入れられる制限内(1MB)であること
最初のトランザクションのみがcoinbaseトランザクションであること
ブロックに含まれるすべてのトランザクションが，「独立したトランザクション検証」のチェックリストをすべて満たすこと

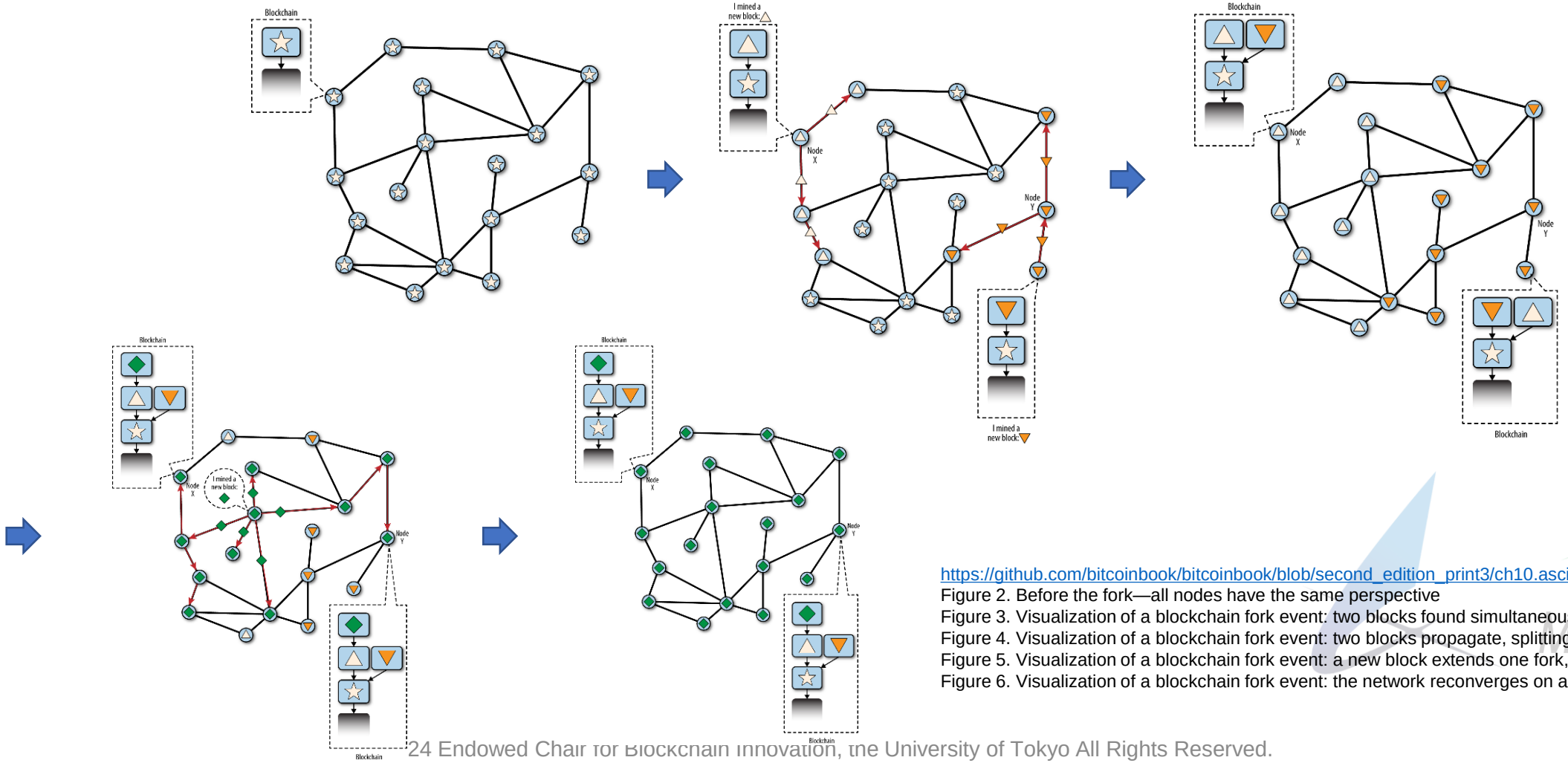
- この検証を全ノードが行う．
  - もし受け取ったノードがブロック内の情報を書き換えたとしても他のノードでの検証に失敗する．
  - もし受け取る一瞬前に自分もマイニングに成功していた場合
    - メインチェーンではなくセカンダリーチェーンにつなげる（次ページ）

ブロックのチェーンの組み立てと選択

- ブロックの検証後，チェーンへの追記が試行される.
  - 無効なブロックは破棄される.
- 各ノードは3つのブロックセットを持つ
  - メインチェーン
    - 最大累積ディフィカルティを持つ（最長チェーン）
  - セカンダリチェーン
    - メインチェーンから分岐した兄弟チェーンも，将来的にメインチェーンに置き換わる可能性があるため保持される
  - オーファンブロックプール
    - 親が不明なブロック
- ブロック内のPrevious Block Hashを見てどのチェーンに属するものか判断する
  - 多くの場合はメインチェーンの最新ブロックに.
- セカンダリチェーンを伸ばすブロックを受け取り，結果としてメインチェーンが置き換わることもある
  - 最長チェーンが入れ替わる
- 前ブロックが見つからないが検証を通るブロックを受け取ると、前ブロックの情報を受け取るまでの間オーファンブロックプールへ保存される
  - ほぼ連続で2つのブロックのマイニングに成功した際，ブロック受け取り順序が逆になった際に生じる

ブロックチェーンフォーク
ブロックチェーンが2つに分かれ，再度1つが選択されるまで

- 伝達ラグによってフォークがおこる．世界中に存在するノードで同時に伝搬を完了させるのは不可能．



https://github.com/bitcoinbook/bitcoinbook/blob/second_edition_print3/ch10.asciidoc
Figure 2. Before the fork—all nodes have the same perspective
Figure 3. Visualization of a blockchain fork event: two blocks found simultaneously
Figure 4. Visualization of a blockchain fork event: two blocks propagate, splitting the network
Figure 5. Visualization of a blockchain fork event: a new block extends one fork, reconverging the network
Figure 6. Visualization of a blockchain fork event: the network reconverges on a new longest chain

マイニングプール
複数人でマイニングを分担する仕組み

- 複数人のマイナーを集めて一緒にマイニングを行い，報酬を分配する
- マイナーは報酬額は減るが，日々平均的な報酬を得られるようになる．

- マネージドプール
  - 管理者がいるマイニングプール
  - 管理者がプールマイニングプロトコルの管理や，フルノードのメンテナンスなどを行う．
  - 報酬は管理者が受け取って，参加者に分配する．
  - 管理者は，報酬の一部を手数料として受けとる．

- P2PPool
  - マネージドプールは管理者によって不正がなされる可能性もある．
  - P2PPoolでは，シェアチェーンと呼ばれる小型ブロックチェーンのようなチェーンを用いて参加者はシェアチェーンを掘る．
    - ビットコインブロックチェーンより低いDifficultyが設定される
  - そのうち，ビットコインブロックチェーンでのDifficultyを充足させるブロックが発見されたらそれを採用する．
  - 報酬の分配ルールもシェアチェーンで管理され特定の誰かが不正できなくなっている．

コンセンサス攻撃

- 特定のマイナーが非常に高いハッシュレートを持っていると，先端のブロック伸長をある程度操作することが可能になる.
  - 2ブロック分マイニングできるくらいのハッシュレートを持っていれば，1ブロック分のフォークは無効化できる.

- 51%アタックと呼ばれる
  - 6ブロック以上を無効化できる可能性がある
  - 51%というのは必ずしも総ハッシュレートの51%が必要というわけではない.

- マイニングプールが不正を行おうとするとかなり問題.

補足）各ノードにて，ブロックチェーンに関するすべてのデータが検証可能

- それぞれのノードは独立に動いている．
  - どこかのノードが◯ブロックまでのブロックチェーンを持っているからといって，そのブロックチェーンを信用して自分のものと取り替える，ということはしない．
  - 新しいトランザクションにせよブロックにせよ，他のノードから受け取ったものは必ず自分自身でその正しさを検証できる．
  - そして，正しいもののみを取り込む．

- このように，他のノードの状態を見ることなく，各ノードは自分自身で各々のパーツを検証していきブロックチェーンを構築する．
  - 最長になるブロックチェーンはこれだ！と自分自身のみで認識する．
    - 他のノードでも同じように各々が持っているブロックチェーンデータは自分で検証する．
  - 先端部分がフォークすることはままあるにせよ，このように各々のノードで保持されるブロックチェーンは結果的に世界中で一致することになる．

まとめ

- 新しいブロック（トランザクション）が正しいといかにしてコンセンサスを得るか
  - 皆が同じルールに基づいて検証を行っている
  - もしこの検証を通らないブロックを伝搬させようとしても他の多くのノードで無効と判断される.
  - 最長ブロックチェーンが残っていくため，いち早く正しく長いチェーンを作ることがマイナーのインセンティブになっている.

- マイニング
  - PoWの解を見つければ報酬を得られる

  - マイニング報酬を得たく，計算コストを損したくないがためにすべてのノードは「正しい」振舞いをする
    - 正しくブロックを作る
    - 正しくブロックを検証する

- コンセンサスアタック
  - ハッシュレートの多くを一部の人が持ってしまうと先端のブロックのコントロールが可能になってしまう. （51%攻撃）
  - マイニングプールがそれができる傾向があるが，P2PPoolなど管理者不在の方法も考案されている

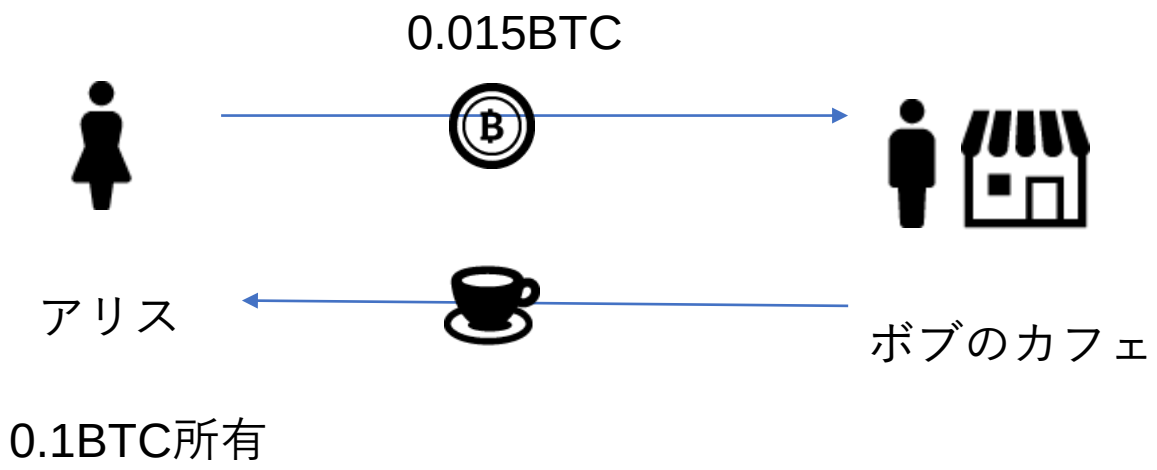
ビットコインの仕組みまとめ

概要

- 初回にあったアリスがボブのカフェで送金する例をもう一度おさらいする.
- このときビットコインはシステム的にどのように動作しているか.
  - トランザクション作成
  - ビットコインネットワークにブロードキャスト
  - マイニングによるブロック生成，伝搬

コーヒー代金の支払いをビットコインで行う

アリスはボブのカフェでコーヒーを注文した.
アリスはもともと0.1BTC持っていた.
コーヒーの代金として0.015BTCをボブに支払った.



トランザクションを作成・ブロードキャスト

- アリスのウォレットはフルノードを仮定.
- 以下の手順でトランザクションを生成する.
  - アリス自身のUTXOを選びインプットとする
    - UTXOのロッキングスクリプトに対して，自身の秘密鍵からアンロッキングスクリプトを生成
  - ボブへの送金アウトプットを作成
    - ボブのアドレスから公開鍵のハッシュを抽出
    - ロッキングスクリプトを生成
  - トランザクション手数料を設定する
  - おつりを自分への送金アウトプットとして作成
    - 自分のアドレスへの送金，ロッキングスクリプト作成

- アリスは，トランザクションをビットコインネットワークにブロードキャストする
  - いくつかのノードに送信
  - 各ノードではトランザクション検証項目リストをもとにこのトランザクションを検証し，トランザクションプールに入れる.
  - そして別のノードにも伝搬する

マイニング，ブロックの伝搬

- マイニング
  - トランザクションプールから手数料がいいものをブロックに入るだけ選択
    - この中でアリスのトランザクションが選ばれたとする
  - コインベーストランザクションの生成
    - 自身へのマイニング報酬
  - ディフィカルティの決定
    - 2016ブロック(14日間)ごとに更新
  - ナンス値を求める
    - 繰り返しハッシュ値の計算を行う
  - マイニングに成功し，ブロックが生成できた.

- 新しいブロックのブロードキャスト
  - いくつかのノードに送信
  - 受け取ったノードはブロックの検証項目リストをもとに検証，伝搬
  - メインチェーンの先端ブロックとして登録

- ボブのウォレット
  - フルノードのウォレットを仮定.
  - 他のフルノードより伝搬されてきた新しいブロックを検証し，自身のブロックチェーンに取り込む.
  - その中に，アリスから自分宛てのトランザクションが含まれていることを確認する.

- 本スライドの著作権は，東京大学ブロックチェーンイノベーション寄付講座に帰属しています．自己の学習用途以外の使用，無断転載・改変等は禁止します．
- ただし，
  - Mastering Bitcoin 2nd edition https://github.com/bitcoinbook/bitcoinbook/tree/second_edition_print3
  - ビットコインとブロックチェーン --- 暗号通貨を支える技術 （著：Andreas M. Antonopoulos 訳： 今井崇也，鳩貝 淳一郎）
- を使用した部分の使用のみ，CC BY-SA 4.0 https://creativecommons.org/licenses/by-sa/4.0/deed.ja のライセンスを適用するものとします．