ブロックチェーンにおけるブルームフィルタを用いたブロック生成通知の最適化

長谷川　毅† 　櫻井　晶†† 　首藤　一幸†

† 京都大学
†† 東京工業大学

あらまし　本稿では，ブロックチェーンにおいてフォーク発生率を抑えるために，ブロックの生成通知を素早く全ノードに届ける手法を提案する。ブロックに含まれるトランザクション群の情報をブルームフィルタを用いて表現し，それを通知として伝播させることで，通知を受け取ったノードは次のブロックのマイニングに移ることが可能になる。シミュレータを用いた実験において，50%ile の伝搬時間は既存手法の 41.1%，90%ile の伝搬時間は既存手法の 39.2% となり，平均伝搬時間から計算されるフォーク発生率は既存手法の 40.8%となった.

キーワード　ブロックチェーン，ブロック伝搬，フォーク率

1 はじめに

ブロックチェーンは非集中性と改竄の困難性を持つ分散システムであり，このことから暗号通貨の基盤として用いられている．現在，Bitcoin [1] に代表される多くの暗号通貨は Proof of Work を利用しているが，この合意アルゴリズムは一定時間に少量のトランザクションしか承認することしかできないという問題がある．具体的には Bitcoin のトランザクション承認性能は，当初，7 TPS（トランザクション / 秒）しかなかった．

トランザクション承認性能が不足している問題を解決する方法として，ブロックの生成間隔を短くする方法が考えられるが，この方法はブロックチェーンのセキュリティを犠牲にする [3]．これは安易にブロックの生成間隔を短くすると，生成されたブロックがブロックチェーンネットワークの全ノードに行き渡る前に，そのブロックが届いていないノードが新たにブロックを生成してしまい，ブロックチェーンに分岐を生じさせてしまう可能性が上昇するためである．このブロックチェーンに分岐が生じることをフォークと呼ぶ．そこでブロックの伝搬時間を短くすることで，フォーク発生率を抑えることができ，ブロックの生成間隔を短縮することが可能になる．

本研究では，ブロックが生成されたことを知らせるブロック生成通知を伝搬させることで，フォークの発生率を低下させる手法を提案する．また全ノードが確率的にブロックを生成するためにフォークが発生し得るブロックチェーンを対象とする．本論の構成は以下の通りである．まず，2 章ではブロックのノード間の伝搬においての関連研究について述べる．3 章では本研究の提案手法について述べる．4 章では評価実験の手法及び結果について示し，考察を述べる．5 章で結論を述べる．

2 関連研究

ブロックを生成したあるノードは，隣接ノードに生成したブロックを送信する．受信したノードはそのブロックを検証し，問題なければ自身の隣接ノードにブロックを転送する．このノード間の転送プロトコルについて，まず通常のノード間のプロトコル (レガシープロトコル) を説明し，Compact Block Relay (CBR) [4] と Graphene [5] を説明する．これらはノード間のブロックの転送時間を短くするためのプロトコルである．

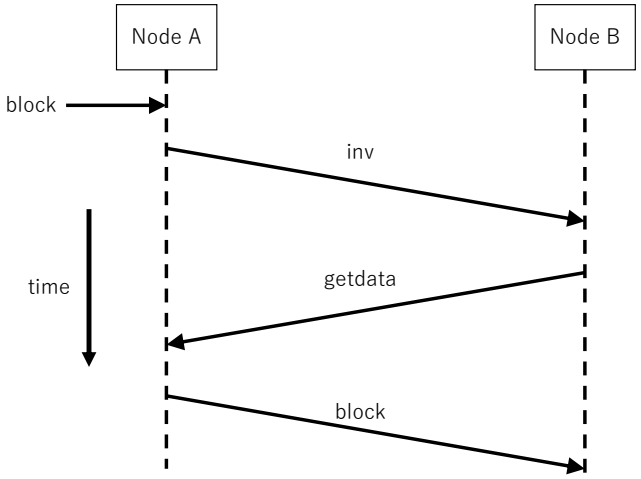

図 1　Bitcoin におけるレガシープロトコルのノード間のやり取り

2.1 通常のブロック伝搬（レガシープロトコル）

レガシープロトコルでは，ノードはブロックに含まれているトランザクション全体を伝搬させる．転送可能なブロックサイズは 1 MiB ほどに制限されている [6].

図 1 にレガシープロトコルのノード間のやり取りを示す．ノード A はブロックを受け取り検証したあと，`inv` メッセージを自身の隣接ノードであるノード B に送る．`inv` メッセージにはブロックのハッシュ値が含まれている．`inv` メッセージを受け取ったノード B は，ブロックのハッシュ値を使い，自身が該当のブロックを持っていなかった場合は `getdata` メッセージをノード A に送りブロックを要求する．`getdata` メッセージを受け取ったノード A は，トランザクション全体を含んだブロックをノー

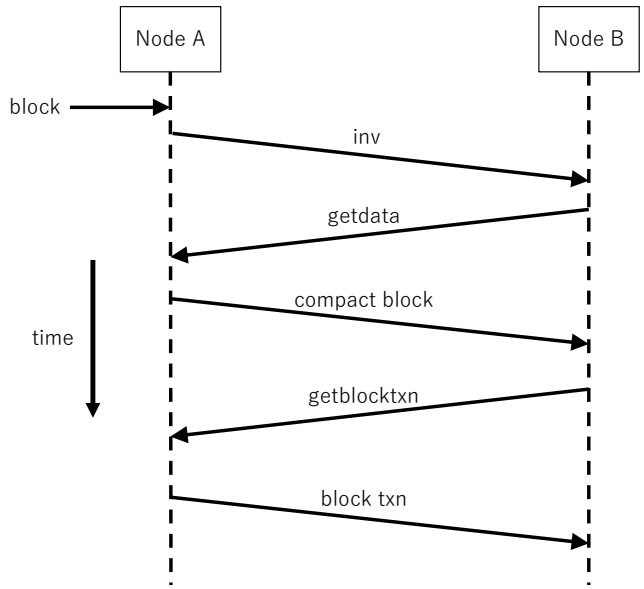

図 2　CBR のノード間のやり取り

ド B に送信する．

2.2　Compact Block Relay (CBR)

Compact Block Relay (CBR) [4] はレガシープロトコルと違い，ブロックに含まれるトランザクション全体ではなく，ブロックヘッダとブロックに含まれるトランザクションの ID からコンパクトブロックを作成し，それを伝搬させる．コンパクトブロックを受け取ったノードは，自身のトランザクションプール (mempool) から該当のトランザクションを取り出しブロックを再構成する．

図 2 に CBR でのノード間のやり取りを示す．ノード A はブロックを受け取り検証したあと，`inv` メッセージを自身の隣接ノードであるノード B に送る．ノード B は該当のブロックを持っていなかった場合 getdata メッセージでブロックを要求する．CBR ではノード A は `getdata` メッセージを受け取ると，ブロックヘッダとブロックに含まれるトランザクションの ID をコンパクトブロックとしてノード B に送る．コンパクトブロックを受け取ったノード B は自身のトランザクションプールから該当のトランザクションを取り出しブロックを再構成する．この際，該当のトランザクションの中に自身のトランザクションプールに存在しないトランザクションが含まれていた場合，ノード A に `getblocktxn` メッセージを送り，足りないトランザクションを要求する．`getblocktxn` メッセージには足りないトランザクションの ID が含まれる．`getblocktxn` メッセージを受け取ったノード A は足りないトランザクションをノード B に送り，それによりノード B はブロックを再構成することが可能になる．

コンパクトブロックは 18 KiB ほどであり [7]，通常のブロックが 1 MiB ほどであることに比べると小さい．受信側のトランザクションプールのトランザクションが足りなければ追加のやり取りが発生するが，レガシープロトコルよりも短い時間で全ノードに行き渡る．

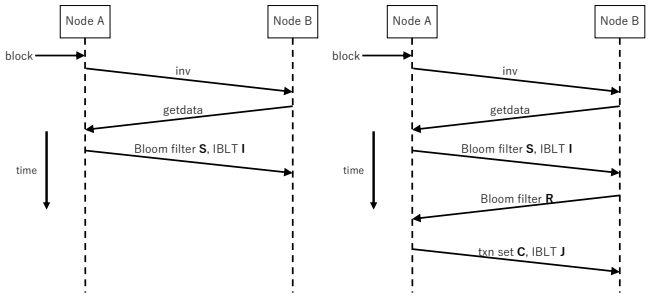

図 3　Graphene のノード間のやり取り

2.3　Graphene

Graphene [5] はブルームフィルタ [8]，Invertible Bloom Lookup Table (IBLT) [9] を用いたプロトコルである．ブルームフィルタは，ある要素が集合に含まれているかどうかを判定できる確率的なデータ構造である．ただし，誤って，含まれていない要素を含まれていると判断してしまう偽陽性の可能性がある．IBLT はブルームフィルタと同じく確率的なデータ構造であり，IBLT としてエンコードした集合の情報を相手に送ることで，相手は自身の集合との差分を求めることができる．この際 IBLT を受けとった側は自身の集合を IBLT として encode し，それを用いて受け取った IBLT を decode する必要があるが，この decode は失敗する可能性がある．この可能性は 2 つの集合の差分が大きいほど高くなる．

図 3 に Graphene でのノード間のやり取りを示す．まずブロックを受け取ったノード A はレガシープロトコルと同じように inv メッセージを隣接ノード B に送る．ノード B は該当のブロックを持っていなかった場合，`getdata` メッセージをノード A に送る．この際自身のトランザクションプールのトランザクションの数も合わせて送る．ノード A は `getdata` メッセージを受け取ると，ブロックに含まれるトランザクションの ID を要素としてブルームフィルタ S，IBLT I を作成し，ノード B に送る．この際ブロックに含まれるトランザクション数，ノード B のトランザクションプールのトランザクション数を考慮してブルームフィルタと IBLT のサイズを決定する．ノード B は自身のトランザクションプールのトランザクションを受け取ったブルームフィルタに通し，陽性集合 Z を得る．そして Z を要素として IBLT I' を作成し，I と I' を用いてブロックに含まれるトランザクションと陽性集合 Z の差分を求める．差分が求められればノード B はブロックに含まれるトランザクションを把握できるため，自身のトランザクションプールからブロックを再構成する．足りない場合は CBR と同じように `gettxn` メッセージで足りないトランザクションをノード A に要求する．

IBLT の decode が失敗し，差分が求められない場合が存在する．その場合はノード B は陽性集合 Z を要素として新たにブルームフィルタ R を作成し，ノード A に送る．ノード A はブルームフィルタ R をもとにノード B に足りないトランザクションセット C を推定する．また R の陽性集合と C の和集合を要素として IBLT J を作成し，C と J をノード B に送る．ノード B はブルームフィルタ S の陽性集合 Z と，トランザクショ

表 1　ブロック生成通知 (ブロックヘッダ) の構造

要素	説明
Transaction filter	ブロック内のトランザクションで使われた UTXO を要素としたブルームフィルタ
前提ブロック情報	前提とした，有効化するブロックの情報
Version	ソフトウェア / プロトコルバージョン番号
Previous Block Hash	親ブロックのハッシュ値
Merkle Root	ブロックの全トランザクションに対するマークルツリーのルートハッシュ
Timestamp	ブロックの生成時刻
Difficulty Target	ブロック生成時の difficulty
Nonce	Proof of Work で用いるカウンタ

ンセット C の和集合を要素として IBLT J' を作成し，J と J' を用いて差分を求め Z を修正する．この際ブロックに含まれるトランザクションをすべてノード B が持っていればブロックを再構成する．持っていなければ CBR と同じようにノード A に要求する．また J と J' を用いた decode が失敗した場合は，それをノード A に知らせ，レガシープロトコルと同じように通常のブロックを伝搬させる．

Graphene においてノード間で転送されるデータのサイズはすべて合わせても CBR よりも少なくなる．ただし decode が失敗したりした場合追加のやり取りが多くなる．

3 提案手法

ブロックチェーンにおいてのフォーク発生率を抑えるための本研究の提案手法について述べる．

本研究では，ブロックが生成されたことを素早くネットワーク上のノードに通知する手法を提案する．以降この通知のことをブロック生成通知と呼ぶ．ブロック生成通知には生成されたブロックの情報を含め，通知を受け取ったノードはその通知をもとにして次のマイニングを開始する．ブロック生成通知が素早く伝搬し，それを受け取ったノードが通知に対応するブロックを親とした次のマイニングに入ることで，フォーク発生率を抑えることができる．ゆえに，空間効率の良いデータ構造であるブルームフィルタを用いて生成されたブロックの情報を表現し、ブロック生成通知に含める．

表 1 にブロック生成通知の構造を示す．本研究では，ブロックのブロックヘッダを拡張し，ブロック生成通知とする．構造としては，トランザクションフィルタ，前提ブロック情報以外は Bitcoin のヘッダと同様である．マイナーは拡張したヘッダを用いてマイニングを行い，ブロックの生成に成功するとヘッダをブロック生成通知として伝搬させる．このようにすることで，ブロック生成通知を受け取ったノードは，生成されたブロックのメタデータの確認，またブロック生成通知の作成者が確かに Proof of Work のマイニングに成功したことを確認することができる．トランザクションフィルタについては 3.1 節で，前提ブロック情報については 3.2 節で詳しく述べる．

図 4 は提案手法の概要を示したものである．

ブロック生成通知を Bitcoin に実装する場合は，ブロック生成通知をマイナーが作るトランザクションに含めブロックチェーンネットワーク上に伝搬させる手法が考えられる．

3.1 トランザクションフィルタ

トランザクションフィルタ F_h は，生成されたブロック B_h 内のトランザクションで消費された UTXO を要素としたブルームフィルタである．ブロック生成通知 N_h を受信したノードは，自身のトランザクションプール内のトランザクションがそれぞれ消費している UTXO をトランザクションフィルタ F_h を用いて判定し，各トランザクションについて，消費する UTXO がすべて陰性だった場合，生成されたブロック B_h には含まれていないと判断し，次のマイニングに含める．

ブルームフィルタには偽陽性の可能性が存在するため，それの対処については 3.2 節で述べる．

3.2 ブロックの有効化

提案手法においてのブロックの有効化についての説明をする．

ブルームフィルタには偽陽性の問題がある．そのことからあるトランザクションフィルタによって偽陽性になったトランザクションはマイニングに含められなくなってしまう．ここで，ある高さ h のトランザクションフィルタ F_h で偽陽性となっているトランザクションは，対応するブロック B_h により，偽陽性であると判断できる．言い換えると，トランザクションフィルタ F_h は，ブロック B_h により無効化できる．ブロック B_h の存在を保証し，対応するトランザクションフィルタ F_h を無効化することをブロック B_h の有効化と呼ぶ．マイニングの際に各ノードが持っているブロックを有効化することで，それに対応するトランザクションフィルタを無効化し偽陽性の問題を解決できる．

ただしブロックの伝播状況は各ノードで一意ではない．それに対し，各ノードはあるブロック B_h を検証する際に，ブロック B_h の生成時に有効化されていたブロックを把握することが必要である．ゆえに提案する手法として，各ノードがマイニングの際所持しており，対応するトランザクションフィルタを無効化したブロックの情報をブロック生成通知に含ませる．これを前提ブロック情報という．高さ h のブロック生成通知 N_h の前提ブロック情報によりブロックを有効化することで、高さ h の時点での有効なブロックの情報を全ノードで一意に保つことができる．

また悪意のあるノードにより，高さ h のマイニングに成功した際，ブロック生成通知 N_h のみを伝搬させ，ブロック B_h を伝搬させない攻撃が考えられる．この攻撃によりブロック B_h が伝搬させられないためそれに対応するトランザクションフィルタ F_h を無効化することができず，F_h により陽性となっているトランザクションがマイニングに含められなくなる．この攻撃に対処するためにブロックの有効化に期限を設ける．具体的には連続して b ブロックの間ブロック B_h が有効化されなかった場合，つまり高さ $h+1$ から $h+b$ のブロック生成通知の前提ブロック情報にブロック B_h が含まれなかった場合，トラン

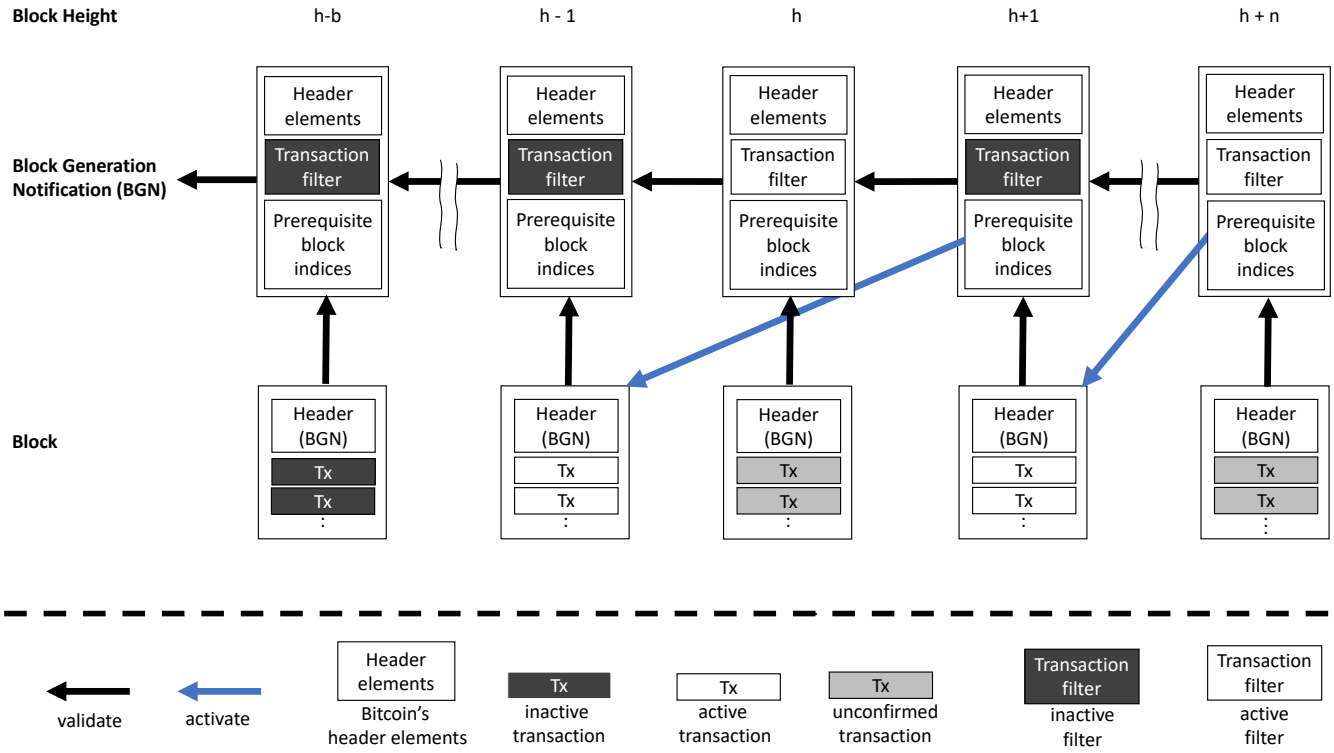

図 4　提案手法の概要

ザクションフィルタ F_h を無効化し，これ以降 B_h を有効化できないようにする．ブロック B_h は有効化ができなくなるため，トランザクションフィルタ F_h で陽性となっていたトランザクションをすべてマイニングに含める．b はシステムワイドなパラメータで，これの決め方については 3.6 節で述べる．

3.3　ブロック生成通知の検証

アルゴリズム 1 は高さ h のブロック生成通知 N_h を受信したノードが行う，ブロック生成通知の検証アルゴリズムである．

アルゴリズム 1 ブロック生成通知 N_h の検証

1: 一つ前のブロック生成通知 N_{h-1} の検証
2: N_h が有効化するブロックの検証
3: ブロック生成通知 N_{h-1} 自体の検証 (Merkle root の検証はしない，それ以外は Bitcoin のヘッダの検証と同じ)

ブロック生成通知 N_h の検証において，まずひとつ前のブロック生成通知 N_{h-1} が検証されていることを確認する．これにより再帰的に以前のブロック生成通知がすべて検証されていることが必要となる．そして次に N_h 内の前提ブロック情報を確認し，有効化されるブロックをそれぞれ検証する．ブロックの検証については 3.4 節で述べる．仮に N_h の受信側が有効化されるブロックを持っていなかった場合は，N_h の送信側に該当のブロックを要求する．最後にブロック生成通知 N_h 自体の検証を行う．この際 Merkle root の検証は行わない．それ以外の検証などは Bitcoin のヘッダの検証と同様である．またブロック B_h の検証は行わない．

3.4　ブロックの検証

アルゴリズム 2 は高さ h のブロック B_h の検証アルゴリズムである．あるノードが高さ $h+1$ 以上のブロック生成通知を受信し，その通知によってブロック B_h が有効化されるとき，B_h の検証が行われる．

アルゴリズム 2 ブロック B_h の検証

1: ブロック生成通知 N_h の検証
2: ブロック B_h 内のトランザクションが統合フィルタ T_h を通過するかの検証
3: ブロック B_h 内のトランザクションの検証

アルゴリズム 3 統合フィルタ T_h の生成

1: 統合フィルタ T_h = ブルームフィルタの初期値
2: $r = h$
3: **while** $r \geqq 0$ **do**
4:　　**if** F_r が有効 **then**
5:　　$T_h = Unite(T_h, F_r)$　▷ $Unite$ は各ビットの論理和を取る関数
6:　　**end if**
7:　　$r--$
8: **end while**
9: **return** T_h

ブロック B_h の検証において，まず対応するブロック生成通知 N_h の検証を行う（アルゴリズム 1）．そして次にブロック内のトランザクションが統合フィルタ T_h を通過するかの検証を行う．統合フィルタ T_h は，高さ h の時点の無効化されていな

アルゴリズム 4 マイニング

1: ブロック生成通知 N_h の検証
2: 手元にある b ブロック前までのまだ有効化されていないブロックに対応するトランザクションフィルタを無効化
3: 手元にある b ブロック前までのまだ有効化されていないブロックを，前提ブロック情報に含める
4: N_h に対する統合フィルタ T_h を生成
5: トランザクションプールに含まれるトランザクションを T_h でフィルタリング
6: ブロックに入れるトランザクションを決める
7: トランザクションフィルタ F_{h+1} を作成
8: トランザクションフィルタと前提ブロック情報をブロックヘッダに含めてマイニングを開始する
9: マイニング成功
10: 生成したブロック B_{h+1} のブロックヘッダをブロック生成通知 N_{h+1} として伝搬させる
11: ブロック B_{h+1} を伝搬させる

いトランザクションフィルタを統合したもので，統合フィルタの生成アルゴリズムはアルゴリズム 3 で示す．そして最後にブロック B_h に含まれるトランザクションが正しいものであるかの検証を行う．また，一つ前のブロックであるブロック B_{h-1} の検証は行わない．これは本研究ではブロックはブロック生成通知によって有効化されるものであり，ブロック B_h が生成されたときに前提となったブロックは，ブロック生成通知 N_h で示され，最初に行う N_h の検証の中で検証されるためである（アルゴリズム 1）．

3.5 マイニング

アルゴリズム 4 はブロック生成通知 N_h を受けとったノードが次のマイニングに入る際のアルゴリズムである．

ブロック生成通知 N_h を受け取ったノードが次のマイニングに入る際，まず受け取った N_h を検証する（アルゴリズム 1）．そして次に，手元にあるブロック b ブロック前までのブロックのうち，まだ有効化されていないブロック B_i があった場合，対応するトランザクションフィルタ F_i を無効化し，前提ブロック情報を更新する（有効化するブロックは複数でも良い）．その後，N_h に対する統合フィルタ T_h を生成する（アルゴリズム 3）．その後生成した統合フィルタ T_h を使い，自身のトランザクションプールのトランザクションをフィルタリングする．このときそれぞれのトランザクションが消費する UTXO を統合フィルタ T_h で判定し，少なくとも 1 つ陽性となったトランザクションはマイニングから除く．そして残ったトランザクションからブロックに入れるトランザクションを決め，新しいトランザクションフィルタ F_{h+1} を作成する．そして拡張したブロックヘッダにトランザクションフィルタと前提ブロック情報を含めてマイニングを開始する．マイニングに成功すると，生成したブロック B_{h+1} のブロックヘッダをブロック生成通知 N_{h+1} として伝搬させ，その後ブロック B_{h+1} を伝搬させる．

3.6 トランザクションフィルタのサイズ

ブロック生成通知に含まれるトランザクションフィルタのサイズについて述べる．ブロック生成通知の伝搬時間を最小化するために，トランザクションフィルタはできるだけ小さいサイズにする．

トランザクションフィルタ F_h は対応するブロック B_h が b ブロックの間有効化されなかった場合に無効化される (3.2 節)．このことから有効なトランザクションフィルタは，b ブロック連続でブロック生成通知によるブロックの有効化がされなかった場合に最も多くなり，その数は b 個である．本研究では，b 個のトランザクションフィルタによって偽陽性となるトランザクションと，本来マイニングに使えるはずのトランザクション全体の比が，パラメータ r 以下になるようにブルームフィルタの偽陽性率を設定する．偽陽性率が決まれば，その際の最も小さいブルームフィルタのサイズは，フィルタの要素数を n, 偽陽性率を f とすると $-\frac{n \ln f}{\ln^2 2}$ ビットとなる．

一つのトランザクションが消費する UTXO の数の平均は 3 である [10]．本研究では，この数字を採用する．消費する UTXO のうち，少なくとも 1 つの UTXO が陽性のとき，そのトランザクションは次のマイニングに含めない．よってブルームフィルタの偽陽性率を f とすると，本来使えるトランザクションが偽陽性トランザクションとなる確率 f' は,

$$f' = 1 - (1-f)^3$$

となる．b 個のブルームフィルタによる偽陽性トランザクションと，本来マイニングに使えるはずのトランザクション全体に占める比が r 以下であるために，f' が満たすべき条件は,

$$\sum_{k=1}^{b} f'(1-f')^{k-1} \leqq r$$

である．これにより f が満たすべき条件は

$$f \leqq 1 - (1-r)^{\frac{1}{3b}}$$

となる．偽陽性率 f とブルームフィルタのサイズはトレードオフであり，ブルームフィルタのサイズはできるだけ小さくするために，ブルームフィルタの偽陽性率 f はできるだけ高くする．よって

$$f = 1 - (1-r)^{\frac{1}{3b}} \tag{1}$$

とする．

パラメータ b の決め方について考察する．式 (1) より，r を固定すると，b が大きくなるにつれて f は小さくなる．偽陽性率 f が小さくなるにつれてフィルターサイズは大きくなるため，b が大きくなるにつれてフィルターサイズは大きくなる．よってフィルターサイズをできるだけ小さくするために，許容できる最小の b を求める．提案手法においては，ブロックは後に生成されたブロック生成通知によって有効化される．このことから b を小さくしすぎると，悪意のあるノードが b 回連続でブロックを生成し，ブロックを有効化しないことにより，悪意のない $b+1$ 個前のブロックが無効化されてしまう可能性が高まってしまう．本研究では b は，悪意のあるノードが b 回連続でブロック生成に成功する確率が double spending 攻撃の成功確率より

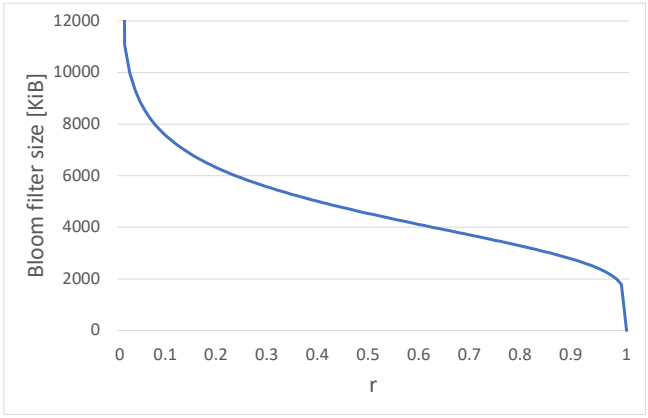

図 5　r を変化させた際のブルームフィルタのサイズ (n=2000, b=4)

表 2　トランザクションフィルタのサイズ (n=2000, b=4)

	r=0.1	r=0.5
filter size [KiB]	7.181	4.391

も低くなるように設定する．悪意のあるノード，またはノード群が全体のハッシュパワーに占める割合を h とすると，それらのノードが b 回連続でブロックの生成に成功する確率は h^b である．これがハッシュレートに応じた double spending 攻撃の成功確率よりも小さくなるように b を設定する．Chaudhary ら [11] によるハッシュレートに応じた double spending 攻撃の発生確率を参考にすると，$b = 4$ で十分である．

パラメータ r について考察する．r を変化させ，$b = 4$，ブロックに含まれるトランザクションの数を 2000 とすると，トランザクションフィルタのサイズは図 5 のようになる．トランザクションの数は [6] を参考にした．図 5 より，r を小さくすればするほど，トランザクションフィルタのサイズは大きくなることがわかる．表 2 は $r = 0.1, 0.5$ のときのトランザクションフィルタのサイズを示したものである．

3.7　攻撃の影響および対策

提案手法への考えられる攻撃とその影響，また，提案手法による対策について述べる．

3.7.1　ブロックを伝搬させない攻撃

3.2 節，3.6 節で述べたように，ブロックを生成した際，ブロック生成通知のみを伝搬させブロックを伝搬させない攻撃が考えられる．これに対処するために b ブロックの間有効化されなかったブロックは対応するトランザクションフィルタを無効化する（3.2 節）．ただし b ブロック積み上がるまで該当のトランザクションフィルタは無効化されないため，一定の間そのトランザクションフィルタによって陽性となったトランザクションがマイニングに含められないという問題がある．本研究では最悪の場合の偽陽性トランザクションの数が一定の割合以下になるようにトランザクションフィルタのサイズを設定する手法を提案している．

3.7.2　ブロックを有効化しない攻撃

この攻撃は，3.6 節で述べたように，悪意のあるノードが b 回連続でブロックを生成し，それらのブロック生成通知でブロックを有効化しないことにより，$b+1$ 個前の悪意のないブロックが無効化されてしまうというものである．この攻撃はパラメータ b が大きくなるにつれて成功確率が低くなるため，本研究では，この攻撃の成功確率が double-spending 攻撃の成功確率よりも低くなるようにパラメータ b の値を設定した．

表 3　ネットワークパラメータ

ノード数	10000
ブロック生成間隔	10 分
ブロックサイズ	1.0 MiB
ハッシュレートの分布	正規分布
ハッシュレートの平均	400000 /sec
ハッシュレートの分散	100000 /sec

3.7.3　特定のトランザクションを陽性に保つ攻撃

ブロックを生成した際，特定のトランザクションが陽性となるようなトランザクションフィルタを作成することで，特定のトランザクションを使わせないようにすることが可能である．これにより，b ブロック積み重なるまでに再びマイニングに成功することで，特定のトランザクションを陽性に保ち，ブロックに含められなくする攻撃が考えられる．この攻撃は b が大きくなるにつれて成功確率が上昇するため，上のブロックを伝搬させない攻撃とトレードオフの関係となる．

3.7.4　不正なトランザクションフィルタを伝搬させる攻撃

すべての UTXO が陽性になるような不正なトランザクションフィルタを伝搬させることで，すべてのトランザクションが使えなくなってしまう．この攻撃へ対処する場合は，仮にブルームフィルタの初期値を 0，陽性ビットを 1 とした場合，1 となっているビットの数の割合に制限をかける手法が考えられる．偶然により悪意のないノードでも制限を超える 1 の数になる可能性もあるため，この場合はブルームフィルタの再構成を許し，再構成回数を示す情報を通知に含める．

4　実験と考察

提案手法がフォークの発生確率を抑えることができることを示すための実験を行う．フォークの発生確率は，ブロックの伝搬時間と密接に関係している．よってシミュレータを使い提案手法や関連研究の伝搬時間を測定する．

4.1　実 験 手 法

実験にはブロックチェーンネットワークシミュレータである SimBlock [12] [13] を用いる．SimBlock には既にレガシープロトコル，CBR の実装がされている．よって追加で Graphene，提案手法の実装を行い，ブロックの伝搬時間を測定した．提案手法については，ブロック生成通知とブロックの伝搬時間を別々に測定したが，他の手法との比較はブロック生成通知の伝搬時間を用いる．これは提案手法では，ブロック生成通知を受信した時点で現在のブロックのマイニングをやめ，次のブロックのマイニングに移るためである．実験は手法ごとのフォーク発生率を明らかにするために行うため，提案手法においてはブロック生成通知の伝搬時間だけを考慮すれば十分である．表 3

表 4　CBR のパラメータ

コンパクトブロックサイズ	18 KiB
チャーンノードの割合	0.97
チャーンノードのブロック再構成失敗率	0.27
コントロールノードのブロック再構成失敗率	0.13

表 5　Graphene のパラメータ

1 ブロックあたりのトランザクション数	2000
ノードのトランザクションプールのトランザクション数	4000
ブロック再構成の際の不足しているトランザクション数	312
Bloom filter S のサイズ	2.434 KiB
IBLT I のサイズ	0.901 KiB
Bloom filter R のサイズ	1.453 KiB
IBLT J のサイズ	1.407 KiB
1 回目の decode 成功確率（シナリオ 1）	100%
1 回目の decode 成功確率（シナリオ 2）	0%
2 回目の decode 成功確率	100%

にシミュレーションで使用するブロックチェーンネットワークのパラメータを示す．表 3 のパラメータ以外のネットワークのパラメータとして，ノード分布，帯域幅，ネットワーク遅延があり，これらは永山ら [7] の値を用いた．

4.1.1　SimBlock における CBR の実装

CBR は SimBlock に実装されているものをそのまま利用する．表 4 に CBR のパラメータを示す．パラメータは [7] を参照した．コントロールノードとはブロックチェーンネットワークに常に参加しているノードで，チャーンノードとはブロックチェーンネットワークへの参加・離脱を繰り返すノードである．チャーンノードとコントロールノードで CBR におけるブロックの再構成失敗率が異なり，ブロック再構成失敗の際に不足しているトランザクション数の分布もコントロールノードとチャーンノードで異なる [14]．永山ら [7] はそれぞれの分布をもとに，不足しているトランザクションを送信する際のコストを計算している．本実験ではそのパラメータをそのまま使用する．

4.1.2　Graphene のモデル化

表 5 に今回実装した SimBlock における Graphene のパラメータを示す．1 ブロックあたりのトランザクション数，ノードのトランザクションプールのトランザクション数は [6] を参照した．ブロック再構成の際にトランザクションが不足している際の不足するトランザクション数の平均は，Imtiaz ら [14] によるとチャーンノードの場合は 312 であり，チャーンノードの際の Bloom filter, IBLT のサイズを計算する際にはこの数値を用いることにした．これらのパラメータから Graphene の Bloom filter，IBLT のサイズを計算した．Bloom filter S, IBLT I は getdata メッセージを受け取った送信ノードが送る Bloom filter, IBLT であり，Bloom filter R, IBLT J は一回目の IBLT の decode が失敗した際の recover するためのものである．その際不足していると思われるトランザクションセット C も伝搬させるが，単純化のために CBR の不足トランザクションのサイズをそのまま流用した．実際には Bloom filter R の偽陽性の問題で CBR の不足トランザクションサイズよりも大きくなる．

表 6　提案手法のパラメータ

ブロック生成通知サイズ	7.272 KiB (r=0.1) or 4.468 KiB (r=0.5)
ブロックのサイズ	1.0 MiB
ブロックの伝搬手法	CBR
有効化ブロックの深さ	1, 2 or 3

表 7　r によるブロック生成通知の伝搬時間

	r=0.1	r=0.5
50%ile 伝搬時間	274 ms	242ms
90%ile 伝搬時間	453 ms	418ms
100%ile 伝搬時間	634 ms	601ms
平均伝搬時間	302 ms	269ms

decode の成功確率はブロック再構成の際の不足トランザクションの割合に基づき決定した．[5] によると 1 回目の decode，つまり IBLT I の decode においては，受信ノードのトランザクションプールに不足しているトランザクション数が，ブロックに含まれるトランザクション数の 2%未満ならば（シナリオ 1），decode はほとんどの場合成功する．不足トランザクション数は [14] をもとにチャーンノード，コントロールノードそれぞれの分布を用い，単純化のために不足トランザクション数の割合が 2%未満の場合，IBLT I の decode は 100%成功するものとした．また [5] によるとその不足トランザクションの割合が 2%以上の際（シナリオ 2），decode はほとんどの場合失敗するとあるので，単純化のためにその場合の decode 成功確率は 0%とした．2 回目の decode，つまり IBLT J の decode の成功確率は，同じく [5] によるとほぼ 100%であるため，必ず成功するものとした．

4.1.3　提案手法の実装

SimBlock に本研究の提案手法を実装した．提案手法のパラメータを表 6 に示す．ブロック生成通知のサイズは 3.6 節で述べたブルームフィルタのサイズに加えて，Bitcoin のブロックヘッダのサイズ 80B を加えたものである．前提ブロック情報は非常に小さいので無視した．偽陽性トランザクションと，本来マイニングに使えるはずのトランザクション全体の比 $r = 0.1, 0.5$ のそれぞれの際の伝搬時間の比較を行う．ブロック生成通知が有効化するブロックの深さは全ノードが一定であるとし，深さ 1 から 3 で伝搬時間の比較を行う．ブロックの伝搬は CBR を用いる．

4.2　実験結果と考察

提案手法のパラメータによる伝搬時間の比較と，レガシープロトコル，CBR，Graphene と提案手法の平均伝搬時間の比較実験の結果と考察を述べる．シミュレーションの終了ブロック高は 100000 ブロックとする．

4.2.1　提案手法のパラメータによる実験結果比較

表 7 は $r = 0.1, 0.5$ の際のそれぞれの伝搬時間の比較である．有効化ブロックの深さは 1 としている．$r = 0.1$ のときのブロック生成通知サイズは 7.433 KiB, $r = 0.5$ のときのブロック生成通知サイズは 4.576 KiB であるため，$r = 0.1$ よりも $r = 0.5$

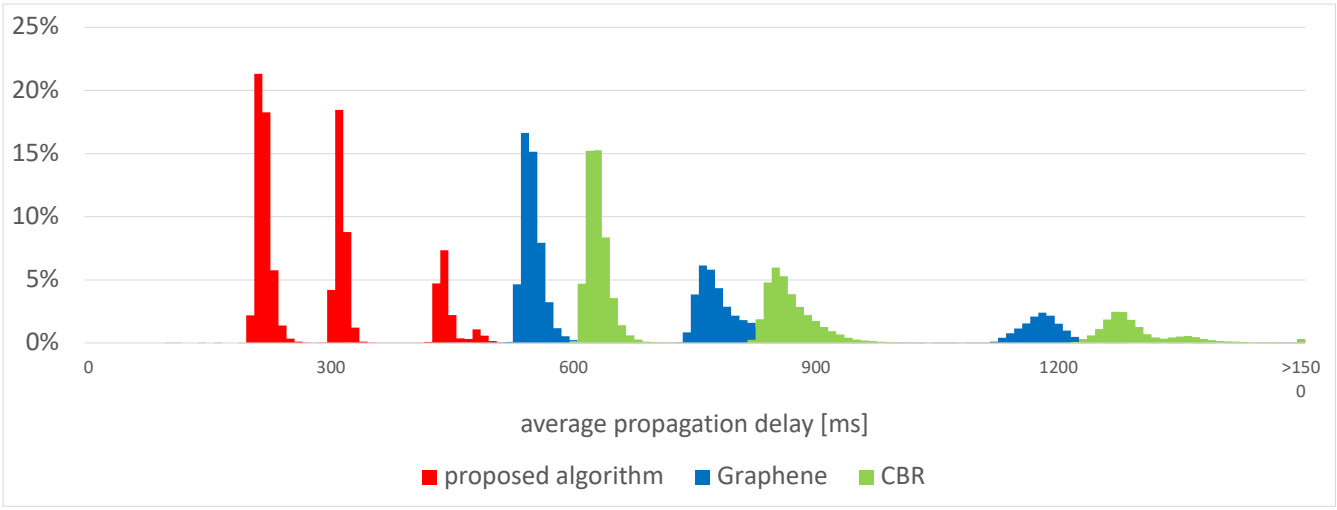

図 6　平均伝搬時間

表 8　有効化ブロックの深さごとのブロック生成通知の伝搬時間

	深さ 1	深さ 2	深さ 3
50%ile 伝搬時間	274 ms	267 ms	267 ms
90%ile 伝搬時間	453 ms	443 ms	443 ms
100%ile 伝搬時間	634 ms	564 ms	570 ms
平均伝搬時間	302 ms	293 ms	293 ms
ブロックの要求回数	3802710	133445	118951

のほうが伝搬時間が小さくなった．

表 8 は有効化ブロックの深さを変化させた際のそれぞれの伝搬時間と，ブロックの要求回数の比較である．$r = 0.1$ としている．表 8 より，有効化ブロックの深さが 1 のときと深さが 2,3 のときでブロックの要求回数に大きな差があるのに対し，有効化ブロックの深さが 2 の場合と 3 の場合の差はあまり大きくない．これは 2 ブロック前のブロックは殆どの場合行き渡っているためであると考えられる．深さ 1 と深さ 2 の場合のブロック生成通知の平均伝搬時間の差は 9ms であり，深さ 2 と深さ 3 の場合に関しては平均伝搬時間は等しかった．有効化ブロック深さが深いほど，該当のブロックが行き渡っているため，ブロックの要求回数は少なくなった．それに伴い平均伝搬時間も短くなった．有効化ブロックの深さは実際には各ノードのブロックの伝播状況によって変化するが，表 8 より，有効化ブロックの深さがいずれであっても伝搬時間の影響は小さかった．

4.2.2　手法間の平均伝搬時間の比較

図 6 に，各手法について，平均伝搬時間のブロック毎頻度を示す．提案手法のパラメータは $r = 0.1$，有効化ブロックの深さは 1 であり，提案手法はブロック生成通知の平均伝搬時間である．提案手法のほぼすべてのブロック生成通知での平均伝搬時間が，CBR, Graphene のブロックの平均伝搬時間よりも小さい．また CBR よりも Graphene のほうがブロックの平均伝搬時間は小さい傾向があった．

図 7，表 9 に，各手法での伝搬時間を示す．提案手法のパラメータは $r = 0.1$，有効化ブロックの深さは 1 であり，提案手法はブロック生成通知の伝搬時間である．提案手法が 50%, 90%, 100%ile，平均のすべての項目で最も小さい平均伝搬時間であった．ついで Graphene, CBR となり，レガシープロトコルが最も大きい平均伝搬時間となった．提案手法の伝搬時間は Graphene と比べて，50%ile は 41.1%，90%ile は 39.2%，100%ile は 27.8%，平均は 40.8%となった．

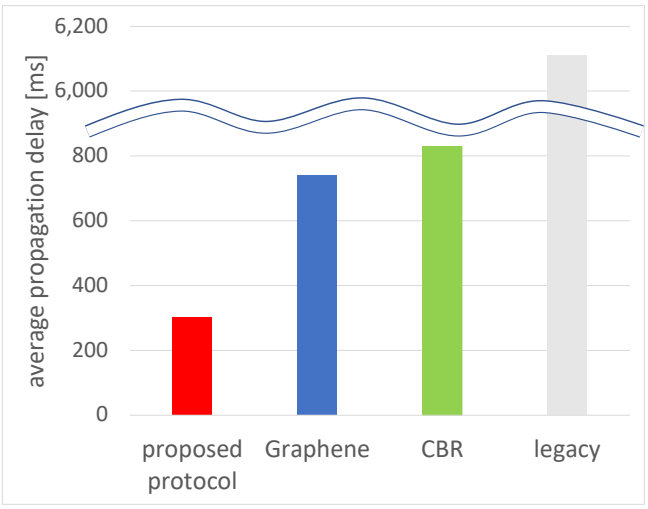

図 7　各パーセンタイルの伝搬時間

表 9　伝 搬 時 間

	提案手法	Graphene	CBR	レガシー
50%ile 伝搬時間	274 ms	666 ms	746 ms	6182 ms
90%ile 伝搬時間	453 ms	1154 ms	1241 ms	8633 ms
100%ile 伝搬時間	634 ms	2282 ms	2367 ms	15326 ms
平均伝搬時間	302 ms	741 ms	828 ms	6478 ms

4.3　ネットワークの通信量

ネットワークの通信量について考察する．図 8 は各手法についての，1 ホップあたりのノード間のブロック伝搬における，メッセージの合計サイズの期待値を示したものである．提案手法はブロックの伝搬には CBR を用いるものとし，CBR は 4.1.1 節で，Graphene は 4.1.2 節で述べたパラメータ，モデル

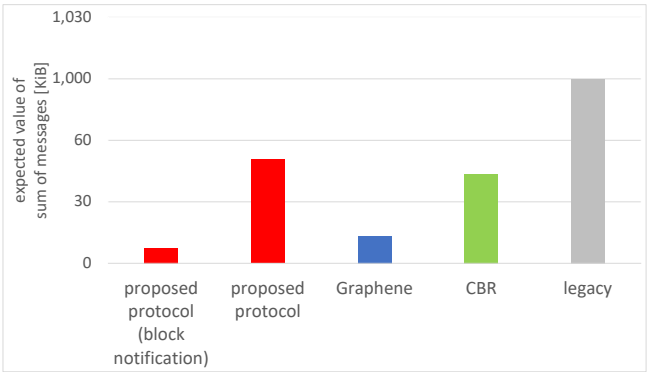

図 8　1 ホップあたりの 2 ノード間のメッセージサイズの期待値

表 10　フォークの発生確率

	提案手法	Graphene	CBR	レガシー
T_W	302ms	741ms	828ms	6478ms
F	0.000503	0.00123	0.00138	0.0107

を元に計算した．提案手法については，ブロック生成通知のみのブロックメッセージと，ブロックとブロック生成通知の合計ブロックメッセージの 2 つを別々に示している．これはフォークの発生確率という観点ではブロック生成通知のメッセージサイズが重要であるためである．1 ホップあたりのノード間のメッセージの合計サイズの期待値は，提案手法のブロック生成通知のみが最も小さく 7.3 KiB であり，ついで Graphene が 13 KiB，CBR が 43 KiB であった．レガシープロトコルは 1 MiB であり，最も大きくなった．これは表 9 から読み取れる伝搬時間を小さい順に並べたものと等しい．また Graphene は 2.3 節で述べたようにノード間のやり取りの数が多くなる傾向があるため，伝搬時間が大きくなっている．これは Graphene と CBR にはメッセージサイズの期待値に大きな差があるのに対し，伝搬時間は大きな差がないことから読み取れる．このことからも提案手法が伝搬時間の観点で優れている理由は，メッセージサイズの期待値が小さいことに加え，ノード間のやり取りが少ないためであると考えられる．

4.4　フォークの発生率

櫻井ら [15] によると，フォーク発生率は次の式で求めることができる．

$$F = \frac{T_W}{T}$$

ここで，F はフォークの発生確率，T_W はノードのハッシュレートで重み付けされたブロックの平均伝搬時間．T はブロックの生成間隔である．表 10 は各手法のハッシュレートで重み付けされたブロックの平均伝搬時間，そしてそこから計算されるフォークの発生確率を示したものであり，図 9 は同様のフォークの発生確率をグラフにしたものである．提案手法の伝搬時間はブロック生成通知を用いている．提案手法のフォーク発生率は Graphene の 40.8%となり、最も低くなった．

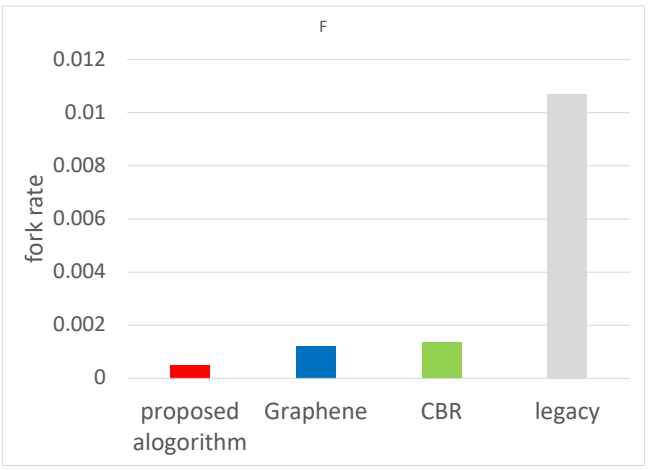

図 9　フォークの発生確率

5　結　　論

本研究では，ブロックの生成通知を全ノードに素早く通知することでフォークの発生確率を抑える手法を提案した．シミュレータを用いた実験において，50%ile の伝搬時間は既存手法の 41.1%，90%ile の伝搬時間は既存手法の 39.2%となり，フォーク発生率の理論値は既存手法の 40.8%となった．ブロック生成間隔を短縮するとフォーク発生率が上昇してしまうが [3]，提案手法によるフォーク発生率の抑制で相殺することで，フォーク発生率を保ったままブロック生成間隔を短縮することもできる．そうした手法も取り組んでいる [16].

謝　　辞

本研究は，京都大学大学院情報学研究科社会情報学専攻 伊藤孝行教授にご支援をいただきました．ここに感謝の意を表します．

文　　献

[1] Satoshi Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Decentralized Business Review (2008) : 21260.
[2] https://en.Bitcoin.it/wiki/Scalability, Accessed: Dec. 10. 2022.
[3] Yonatan Sompolinsky and Aviv Zohar, “Secure high-rate transaction processing in Bitcoin,” in International Conference on Financial Cryptography and Data Security (FC 2015), Springer, 2015, pp. 507–527.
[4] Matt Corallo, 2016, “BIP 152: Compact block relay,” https://github.com/Bitcoin/bips/blob/master/bip-0152.mediawiki.
[5] A. Pinar Ozisik, et al., ”Graphene: efficient interactive set reconciliation applied to blockchain propagation,” Proceedings of the ACM Special Interest Group on Data Communication, 2019, 303-317.
[6] https://www.blockchain.com/explorer/charts, Accessed: Dec. 10. 2022.
[7] Ryunosuke Nagayama, Ryohei Banno, and Kazuyuki Shudo, “Identifying impacts of protocol and internet development on the Bitcoin network,” 2020 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2020.
[8] Burton H. Bloom, 1970, “Space/Time Trade-offs in Hash

Coding with Allowable Errors," Commun, ACM 13, 7 (July 1970) , 422-426.

[9] Michael T. Goodrich and Michael Mitzenmacher, (Sept 2011), "Invertible bloom lookup tables," In Conf, on Comm, Control, and Computing, 792-799

[10] https://Bitcoinvisuals.com/chain-input-count-tx, Accessed: Dec. 10. 2022.

[11] Chaudhary Kaylash, Vishal Chand, and Ansgar Fehnker, "Double-Spending Analysis of Bitcoin," PACIS, 2020.

[12] Yusuke Aoki, Kai Otsuki, Takeshi Kaneko, Ryohei Banno, and Kazuyuki Shudo, " SimBlock: A Blockchain Network Simulator, " in Proc, Workshop on Cryptocurrencies and Blockchains for Distributed Systems (CryBlock 2019, In conjunction with IEEE INFOCOM 2019), April 2019.

[13] Ryohei Banno, and Kazuyuki Shudo, " Simulating a Blockchain Network with SimBlock, " in Proc. IEEE International Conference on Blockchain and Cryptocurrency (ICBC 2019), May 2019, pp. 3-4

[14] Imtiaz, Muhammad Anas, et al. "Churn in the Bitcoin network: Characterization and impact," 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, 2019.

[15] Akira Sakurai, Kazuyuki Shudo, "Impact of the Hash Rate on the Theoretical Fork Rate of Blockchain," Proc, IEEE ICCE 2023, 2023

[16] Masumi Arakawa, and Kazuyuki Shudo, "Block Interval Adjustment Based on Block Propagation Time in a Blockchain," 2022 IEEE International Conference on Blockchain (Blockchain), IEEE, 2022.