博士論文

スケーラビリティに基づくブロックチェーン

インフラストラクチャーの数学的分析

（Mathematical Analysis of Blockchain Infrastructure

based on Scalability）

令和 4 年度

総合分析情報学コース

49-187404

清家大嗣

指導教官 越塚 登 教授

博士学位論文概要 2022 年度（令和 4 年度）

スケーラビリティに基づくブロックチェーン インフラストラクチャーの数学的分析

近年, 決済やスマートコントラクト, 所有権管理やディジタル識別用のインフラストラクチャーとして, ブロックチェーンと呼ばれるトランザクションベースの分散台帳への期待が社会的に高まっている. 一方, トランザクション承認遅延が, ブロックチェーンを利用するアプリケーション開発の妨げとなっている. トランザクションはブロックと呼ばれる単位にまとめられ, 合意形成によってブロックが正しいチェーンに追加されるタイミングで承認される. このため, チェーン成長の振舞いやキューイング処理によって, トランザクション承認に遅れが生じる.

これら 2 つの原因で生じる遅延は, 同時に改善できない. チェーン成長速度 v_{cg} を増加させる場合, 高速でブロックを伝搬させるため, 一つのブロックで処理可能なトランザクションのバッチサイズを減少させなくてはならない. つまり, トランザクションの最短承認時間と, キューイング遅延の間にはトレードオフが存在する. 仮にブロックチェーンシステム内に許容量を超えるトランザクションが流入した場合, トランザクション承認時間は指数関数的に増大する. このことは, ブロックチェーンのスケーラビリティ問題として知られている.

このため, ブロックチェーン外側のデータベース (オフチェーンストレージ) を利用し, トランザクションの送信回数やデータサイズを減少し, 実質的なブロックサイズを増加するオフチェーン技術が提案されている. しかし, 現状ではトランザクション承認遅延を劇的に改善させるまでには至っていない.

このような背景から, ブロックチェーンアプリケーションでは, データベースの参照,

有効期限 (Expiration time) 付きのトランザクションの送信, 処理順序に意味のあるトランザクションの送信時に注意を要する. 例えば, 意図した順番でトランザクションを処理するために, ロック時間 (Lock time) 付きのトランザクションが利用されている.

本論文の目的は, 上記で述べた手続きを実行するアプリケーションのスケーラビリティ, 性能, セキュリティに影響を及ぼす, トランザクション承認時間を数学的に評価する実用的な予測モデルの提案である. このため, チェーン成長の振舞いと, キューイング処理に関する 2 つの理論的分析を行った.

最初に, 実測コストの高いブロック伝搬遅延の上限 D を必要とせずに, チェーン成長の振舞いの時間分布を分析する手法を提案する. 本手法では, 容易に計測可能, かつブロック伝搬時間と相関のあるパラメータであるフォーク確率 P_F を用いる.

次に, オフチェーン技術導入時のキューイング処理をモデル化して, オフチェーンストレージを更新するオフチェーントランザクション (OTX) の待ち時間を数学的に分析する. 本分析では, 最もメジャーなオフチェーン技術の一つである楽観的な Rollup (ORU) を対象とした. この結果, OTX をまとめるバッチサイズ b や, タイムアウト時間 T といったパラメータを理論的に決定可能となる.

また, 3 つの実アプリケーション (ucode の所有権管理, オフチェーンを利用した ucode 所有権割振り, 深層学習モデル生成プロセスの検証) を実装し, トランザクション承認遅延により発生する問題 (例えば, フロントランニング攻撃) を考慮したパラメータ設定を行う事例研究を行った.

以上の研究を通して, トランザクション承認時間を実用的な予測モデル化に貢献した. また, 実アプリケーションで生じる問題の定式化と, 提案評価手法に基づいて対策のためのパラメータ設定を行うことにより, 開発時の留意点の明確化に貢献した.

キーワード： BC (BlockChain), Distributed Computing, Queuing Theory, IoT (Internet of Things), AI (Artificial Intelligence), DL (Deep Learning)

Dissertation Abstract, Year 2022

Mathematical Analysis of Blockchain Infrastructure based on Scalability

In recent years, there has been a growing social expectation for a transaction-based distributed ledger called blockchain as an infrastructure for payments, smart contracts, ownership management, and digital identification. Meanwhile, transaction confirmation delays in a blockchain system make it difficult to develop applications. Transactions are grouped into a block and they are confirmed in a batch manner when the block is added to the correct chain by a consensus. Therefore, the chain growth behaviors and queueing processes cause transaction confirmation delays.

Both delays can't be mitigated at the same time. To increase the chain growth rate v_{cg}, the transaction batch size of a block must be reduced to increase block propagation speed, and vice versa. There exist trade-offs between the minimum transaction confirmation time and the queueing delay. If the transaction arrival rate exceeds a critical value, the mean transaction confirmation time exponentially increases. This is well known as the blockchain scalability problem.

For this reason, blockchain systems adopt offchain technology that uses databases outside of the blockchain to reduce the number of transactions per second and the data size so that the effective block size increases. However, current off-chain technology has not dramatically improved transaction confirmation delays.

Given this background, blockchain applications should care to refer blockchain database, send a transaction with the expiration time, and send transactions that produce differ-

ent results in different processing orders. For example, to confirm transactions in the desired order, a transaction with the lock time is used.

The objective of this dissertation is to propose a practical, predictive model that mathematically evaluates the transaction confirmation time, which affects the scalability, performance, and security of applications that perform the aforementioned procedures. For this goal, I conduct two theoretical analyses based on the chain growth behavior and queueing process. First, I propose a method for analyzing the time distribution of the chain growth interval without an upper bound on block propagation delay, which is expensive to measure. Instead, the method uses a fork probability P_F that is easy to measure and correlates with block propagation delay.

Next, I mathematically analyze the latency of a offchain transaction (OTX) that updates offchain storage by modeling the queueing process for Optimistic RollUp (ORU), which is one of the major offchain technologies. As a result, parameters such as the batch size b for OTXs and the timeout time T can be theoretically determined.

In addition, I implemented three blockchain applications (ucode ownership management, ucode allocation based on offchain technology, and verification of DNN model generation process) and conducted three case studies to set parameters that take into account the problems of transaction confirmation delays, such as front-running attacks.

From two theoretical analyses and three case studies, I contributed to practically modeling the transaction confirmation process in blockchain systems, formulating the problems that arise from transaction confirmation delays, and giving guidelines on how to set parameters for countermeasures based on the proposed predictive model.

Keywords： BC (BlockChain), Distributed Computing, Queuing Theory, IoT (Internet of Things), AI (Artificial Intelligence), DL (Deep Learning)

目 次

図 目 次

表 目 次

第1章　背景

本章では, 最初にブロックチェーン技術に対する社会の期待について触れる. 続いて, ブロックチェーン技術を土台とした分散データベースシステムのスケーラビリティ問題について説明する. ブロックチェーンでは, データベース更新単位のトランザクションが, システム内に許容量を超えて流入すると, 短時間でトランザクション承認できなくなる. また, トランザクション承認時間の数学的評価が, ブロックチェーンインフラストラクチャーを利用したアプリケーション開発において, 重要である理由を述べる. 最後に, 従来のトランザクション承認遅延の評価手法が抱える問題点について指摘し, 本論文の目的を明らかにする.

1.1　ブロックチェーン技術に対する社会の期待

ブロックチェーンは, 2008 年 11 月に提案された暗号通貨 Bitcoin [1] の中核技術であり, 分散データベースを実現する技術の一つである. Bitcoin は, 暗号通貨をグローバルに送金可能にするというミッションクリティカルなシステムでありながら, 分散データベースの一貫性を保証する特定の権威機関を必要とせず, 誰でも自由に Bitcoin の P2P ネットワークに参加可能である. このため, Bitcoin は社会的に大きく注目された. 加えて, Bitcoin はオープンソースソフトウェアであり, システムの仕様, ソースコードが全て GitHub 上でオープンとなっている [2] (2009 年に S. Nakamoto により初めて公開されたバージョン 0.1 は, SourceForge で公開されていた). システムの運用と開発がオープンという性質から, 世界中の多くの人によって Bitcoin ブロックチェーンは実行, 検証, 改善されている. その結果, Bitcoin は信頼性の高い P2P 決済のソースコードを社会に

提供するようになった.

2013 年には, 汎用的な分散データベース更新機能を持つブロックチェーン Ethereum [3] が提案され, 2015 年に稼働開始した. これが, ブロックチェーンのプラットフォーム化の始まりである. Ethereum は, 単純なスクリプトコードしか実行できない Bitcoin とは異なり, チューリング完全なコードを実行可能なシステム [4] である. 言い換えれば, C, Java, Python といった汎用的なプログラミング言語で書かれたコードを実行可能なコンピュータと同等の計算能力を持つ. Ethereum も Bitcoin と同様に, GitHub 上のコミュニティが開発するオープンソースソフトウェアである [5]. Ethereum 以降も, Hyperledger [6], Corda [7], Diem [8] といった数多くのプロジェクトが提案され, 全てオープンなソフトウェアとなっている.

マーケッツアンドマーケッツ社の調査 [9] では, ブロックチェーン市場は 2021 年の 49 億ドルから, 2026 年には 674 億ドルまで拡大すると予測されている. その応用範囲も, 決済, スマートコントラクト, 所有権管理, ディジタル識別といった社会基盤に関わる重要な要素が多い. 以上より, ブロックチェーン技術に対する社会からの期待は非常に高いといえる.

1.2 ブロックチェーンとスケーラビリティ問題

上記のような社会的期待がある一方, ブロックチェーンシステムにはスケーラビリティの問題 [10, 11] が存在する. システム内のトランザクション数が許容量を超過した場合に, トランザクションを短時間で処理できない (トランザクション承認遅延が発生する). このことは, ブロックチェーンの成長速度と, 一つのブロックに保存可能なトランザクション数に制限があることと密接に関係している. 本節では, トランザクション承認遅延の発生メカニズムと, スケーラビリティ問題緩和のために提案されているオフチェーン技術について説明する. また, オフチェーン技術を含んだブロックチェーンシステム全体をブロックチェーンインフラストラクチャーと定義する.

1.2.1 トランザクション承認遅延の発生メカニズム

ブロックチェーンは, 2008 年 11 月に提案された暗号通貨 Bitcoin [1] の土台にもなっている, 分散データベースを実現する技術の一つである. ブロックチェーンシステムは, 信頼できる中央管理者がいない状態でも動作し, コンセンサス (合意形成) アルゴリズムによって, ネットワークに参加する複数ノード間での同一のデータベース共有を可能にする. 同一のデータベースは, 一連のコマンド $c_1, c_2, c_3, ...$ を同順序で実行することで作成される. この手続きは, 状態複製 (state replication) [12] と呼ばれる. ブロックチェーンの分野では, この一つ一つのコマンドを定義した不可分 (Atomic) なデータ単位をトランザクションと呼んでいる.

コマンドを定義したトランザクション群は, ブロックと呼ばれる単位でまとめられる. ブロックは時系列に並べられ, ブロックチェーンと呼ばれる構造を持つ. どのブロックチェーンが正規であるかは, 合意形成により決定される. ブロック内のトランザクションは処理順序が定められており, 起源となるブロックから, 最新のブロック高までに含まれている全トランザクションを順に処理することで, 同一のデータベースを作成できる.

一方, トランザクション群をブロック単位でバッチ処理するため, ブロックチェーンシステムにはトランザクション承認遅延という問題 [13] が生じる. トランザクションの承認とは, 正しいブロックチェーンにトランザクションが含まれたと判断され, ネットワークに参加するノードによってトランザクションが処理されることを指す. トランザクション承認遅延は, 以下の二つの原因で生じる.

- チェーン成長の振舞いによる, トランザクション承認時刻の制約
- ブロック内トランザクション数の限界による, キューイング処理の制約

トランザクションはブロック単位でまとめて処理されるため, トランザクションの承認時刻は, ブロックが正しいとされるチェーンに接続される時刻と厳密に一致する. 従って, 合意形成に基づくブロックチェーン成長の振舞いがトランザクションの承認時刻を決定する. 単位時間当たりに正しいとされるチェーンが伸びる長さを, ブロックチェーン成長

速度 v_{cg} と定義すれば, トランザクションの承認時間は v_{cg} に依存した変数となる.

ブロックチェーンのキューイング処理は, ブロック内のトランザクション数に限界値が設定されているために生じる. 限界値は, ネットワーク上の全ノードにブロックを効率的に伝搬するために設けられており, ブロックサイズと呼ばれる. ブロックを新たに作成し, ブロックチェーンを伸ばすタイミングで, ブロックサイズを超過するトランザクションがシステム内に存在する場合, ブロックに収まらないトランザクションは次以降のブロックに含まれる. つまり, トランザクション処理にキューイング遅延が発生する.

これら二つの原因で発生する遅延は, 同時に改善できない. ブロックチェーンの成長速度 v_{cg} を増加させると, 合意形成のためにネットワーク全体にブロックを高速で伝搬する必要が生まれ, ブロックサイズを減少させなくてはならない (逆もまた然りである). 従って, ブロックチェーンシステムでは, 大量のトランザクションデータの流入時に, トランザクションを短時間で処理できない (トランザクション承認遅延が増加する).

1.2.2 スケーラビリティ問題を緩和するオフチェーン技術と, ブロックチェーンインフラストラクチャーの定義

スケーラビリティ問題を緩和するため, ブロックチェーンの外側を利用するオフチェーン技術 (Offchain technology) が提案されている. オフチェーン技術では, 土台となるブロックチェーンの分散データベース, 暗号技術, 新たに導入するオフチェーンプロトコルを用いて, ブロックチェーン外部 (オフチェーン) のデータの完全性 (Data integrity) を保証する. オフチェーン技術により, ネットワーク全体のトランザクション送信回数またはトランザクションのデータサイズを減少できれば, 実質的なブロックサイズを増加させたことになり, キューイング処理由来のトランザクション承認遅延が改善される. 例えば, 外部データの完全性を保証するために, 暗号学的ハッシュのみをブロックチェーンのストレージに保管し, トランザクションのデータサイズを下げることもオフチェーン技術の一つである.

実装されているオフチェーン技術として, Bitcoin の決済に特化した Lightning Network [14], Ethereum [3] を土台として動作する Arbitrum [15], Ethereum コミュニティで開発が続く楽観的なロールアップ (ORU: Optimistic RollUp) [16] やゼロ知識証明を用いたロールアップ (zk-Rollup: Zero Knowledge Rollup) [17] が存在する. 本研究では, オフチェーン技術を含んだブロックチェーンシステム全体をブロックチェーンインフラストラクチャーと定義する.

1.3 トランザクション承認遅延評価手法の必要性

1.2.1 で詳述したように, ブロックチェーンシステムのトランザクション承認遅延を改善することは本質的に困難である. また, トランザクション承認遅延がアプリケーションの性質に与える影響は大きい. ブロックチェーンネットワークに送信されたトランザクションの処理内容が, 分散データベースに即時に反映されないため, 以下のような処理を実行する際に注意を要するためである.

- データの参照
- 有効期限のある手続き
- 処理順序に意味のある手続き

例えば Bitcoin の場合, トランザクションがネットワークに受信されてから承認されるまでに, 平均 600 秒程度かかるという報告がされている [18]. つまり, ブロックチェーンの分散データベースを参照し, Bitcoin ウォレットアプリケーションの画面に送金結果を反映するには, トランザクションを送信してから 600 秒程度待つ必要がある.

有効期限 (Expiration time) 付きのトランザクションを用いたアプリケーションとして, ENS (Ethereum Name Service) [19] の Vickrey オークション [20] が挙げられる. ENS のオークションでは, 指定されたタイムスタンプの時刻までに, 入札を実行するトランザクションが承認されなくてはならない. つまり, トランザクションの承認遅延を考慮した上で, 入札手続きを進める必要がある.

トランザクションの処理順序により, 作成されるブロックチェーンの分散データベースは変化する[1]. このため, トランザクションを意図した順序で処理するために, ロック時間 (Lock time) 付きのトランザクションが利用されている. トランザクションの Locktime フィールド [21] に Unix タイムスタンプを入力することで, 指定した時刻より前にトランザクションが承認されることはなくなる (タイムスタンプは, 先日付小切手の日付のような役割を果たす). 処理を後に回したいトランザクションのロック時間は, 先に処理したいトランザクションの承認予測時刻よりも遅く設定される. 以上のようにして, トランザクションの処理順序を制御する. ロック時間付きのトランザクションは, Lightning Network [14] を実現するための必須要素として知られている.

このように, トランザクション承認遅延時間はブロックチェーンのアプリケーション設計時に考慮すべき重要なパラメーターである. 特に, 有効期限やロック時間を決定する場合, トランザクション承認遅延を数式により評価する手法が求められる. この手法を確立するために, 本研究では, トランザクション承認遅延の原因であるチェーン成長の振舞いとキューイング処理を数学的に分析する.

1.4 本研究の目的

本研究の目的は, アプリケーションのスケーラビリティ, 性能, セキュリティに影響を及ぼす, トランザクション承認時間を数学的に評価する実用的な予測モデルの提案である. 1.2.1 で述べたように, 分散した複数ノードが合意形成を行うブロックチェーンシステムにおいて, トランザクション承認遅延を改善することは本質的に困難である. また, 4.1 で説明したように, トランザクション処理の制御の実行や, アプリケーションの挙動を予測するためには, トランザクション承認の遅延時間を評価する予測モデルが必要不可欠である.

トランザクション承認遅延は, チェーン成長の振舞いとキューイング処理が原因で発

[1]例として, データベースの初期値 $x = 0$ を操作する, 二つのトランザクション Tx1: $x = x + 1$, Tx2: $x = 2 \times x$ を考える. Tx1 が先に処理されると $x = 2$ となり, Tx2 が先に処理された場合 $x = 0$ となる.

生する (1.2.1 を参照). これら 2 つの理論的分析を行い, 予測モデルを確立した研究は数多く存在する. 例えば, [22, 23] では, Bitcoin で採用されている合意形成アルゴリズム PoW (Proof of Work) のチェーン成長の振舞い分析を行っている. また, [13, 24] では, 独自の待ち行列モデルに基づいて, ブロックチェーンシステムのトランザクション承認遅延を評価している.

しかし, 従来研究では以下の三点を考慮していないという問題がある. これらは, 実アプリケーションに対して予測モデルを適用する際に, 考慮すべき内容である. 本研究では, これら三点を考慮した予測モデルの確立を目指す.

- 実測コストの高いパラメータを用いて, チェーン成長の振舞い分析を行っている
- オフチェーン技術導入時のキューイング処理の数学的分析を行っていない
- 実アプリケーションに手法を適用した場合の評価を行っていない

一点目は, 従来研究 [22, 23] 等で生じている問題である. 上記研究では, ブロック伝搬遅延の上限 D というパラメータを前提としている. D は, 多数のノードを P2P ネットワーク上に配備して, ブロック伝搬時間を計測し続ける必要があるため, 実測コストが高い. そこで, 本研究では D の代わりに, 単独のノードで容易に計測が可能なフォーク確率 P_F を用いて, チェーン成長の振舞い分析手法を確立することを第一目的とする (第 4 章が該当). フォークとは, 同じブロック高に正しいチェーンの候補ブロックが複数存在する事象であり, ブロック伝搬時間の増加により発生確率が増加する. つまり, フォーク確率 P_F を用いた分析により, ブロック伝搬時間の影響を間接的に考慮した, チェーン成長の振舞いモデルを構築する.

二点目は, オフチェーン技術を導入した場合のトランザクション承認遅延を, 数学的に分析した研究が行われてない点である. オフチェーン技術は, アプリケーションレベルで独自なプロトコルとして実装されているケースも多く, トランザクションの承認遅延を一般的に数理的に評価することは困難である. そこで, 本研究では, ブロックチェーンの外側 (オフチェーン) の任意データを更新可能であり, メジャーなオフチェーン技術の一つである ORU (Optimistic RollUp) に着目する.

ORU では, オフチェーンのデータベースを更新するトランザクションを, オフチェーントランザクション (OTX) と呼ぶ. ORU では, 複数の OTX により更新されたデータベースの内容を要約したトランザクションを, 一定周期で土台となるブロックチェーンに送信する. つまり, ORU を導入すると, 従来のブロックチェーンのトランザクションのキューイング処理に加えて, OTX のキューイング処理が発生する. このため, OTX の承認遅延を数学的に評価する手法の確立が必要となる. これを, 本研究の第二目的とする (第 5 章が該当).

三点目は, トランザクション承認時間の評価結果を用いて, 実アプリケーションの開発時のパラメータ設定を行うケーススタディが行われてない点である. 例えば, トランザクション承認遅延によって, フロントランニング [25, 26] と呼ばれる問題が報告されている. フロントランニングとは, トランザクションがブロックチェーンネットワークに送信され承認される間に, その公開情報を利用して先に何らかのアクションを行うことである. トランザクション承認遅延を評価できるようになれば, フロントランニング攻撃に晒されている可能性のある時間を事前に予測評価し, 適切に対策できるようになる.

本研究では, ブロックチェーンインフラストラクチャーを利用した 3 つのアプリケーション (ucode 所有権管理, オフチェーン技術を利用した ucode 割振り, 深層学習モデル生成プロセスの検証) の作成を通じて, トランザクション承認遅延により発生する問題について, 数理的に評価し対策を行う 3 つのケーススタディを行う. これを第三の目的とする (第 6, 7, 8 章が該当).

1.5 本論文の構成

本論文の次章以降の構成は次のようになっている. 第 2 章では, ブロックチェーン技術及び, トランザクション承認遅延を緩和するオフチェーン技術について解説する.

第 3 章では, ブロックチェーンシステムのトランザクション承認遅延を引き起こす, チェーン成長の振舞いとキューイング処理に関連する研究を紹介する.

第 4 章では, PoW ブロックチェーンを例にし, トランザクション承認遅延の一つの原因であるチェーン成長の振舞いに関する理論的な分析を行う. 第 4 章の内容は, 第一著者として発表した論文 [27] をベースとしている.

第 5 章では, オフチェーン技術である ORU (Optimistic RollUp) に関する待ち行列モデルを提案し, 理論的な分析を行う. 第 5 章の内容は, 第一著者として発表した論文 [28] をベースとしている.

第 6 章では, IoT (Internet of Things) 用途の標準化された数値 ID である ucode (ubiquitous code) の所有権管理をブロックチェーン上で行うシステムの提案, 実装, 分析を行った. 本章では, フロントランニング攻撃の対策に用いられる, ロック時間付きトランザクションが承認される際に満たすべき不等式評価を, 第 4 章の分析結果を利用して行った. 第 6 章の内容は, 第一著者として発表した論文 [29] をベースとしている.

第 7 章では, オフチェーン技術を用いた ucode の割振りを可能にする手法の提案, 実装, 分析を行った. 本章では, 第 5 章の ORU 分析と同様な待ち行列モデルを考えることで, バッチサイズ b, タイムアウト時間 T のパラメータを調整することで, ucode 割振りを実行する OTX の待ち時間 $E[W_{ur}]$ を評価可能となった. 第 7 章の内容は, 第一著者として発表した論文 [28] をベースとしている.

第 8 章では, DNN (Deep Neural Network) のモデル生成プロセスをブロックチェーンの改ざん耐性を利用した上で, 事後的に検証可能にするシステムの提案, 概念実装, 分析を行った. 本章では, ORU の紛争解決プロトコル (2.4.1 を参照) で用いられる, 有効期限付きトランザクションが承認される際に満たすべき不等式評価を, 第 4 章の分析結果を利用して行った. 第 8 章の内容は, 第一著者として発表した論文 [30] をベースとしている.

第 9 章において, 本論文の結論及び, ブロックチェーンインフラストラクチャーの将来の課題について述べる.

第2章　ブロックチェーンとオフチェーン技術について

本章では, ブロックチェーンインフラストラクチャーの要素技術であるブロックチェーン技術, 及びオフチェーン技術について解説する.

2.1　ブロックチェーン技術

ブロックチェーンは, ディジタル通貨 Bitcoin [1] の土台技術としてよく知られている. ブロックチェーンでは, コンセンサス (合意形成) アルゴリズムを用いて, P2P ネットワークの複数参加ノードが同一のデータベース共有を行う. Bitcoin ブロックチェーンでは, ディジタル通貨の取引情報に関するデータを, 複数ノード間で共有している.

同一のデータベース共有を可能にするメカニズムを, 以下に示す. 各ノードは, 合意形成により, ブロック高 0 から最新のブロック高まで同一なブロック列 (ブロックチェーン) を共有する. 各ブロックは, データベースの更新ルールが記述されているトランザクションを複数含み, その処理順はユニークに定まっている. 合意したブロックチェーンについて, ブロック高 0 からトランザクションを順番に処理してデータベースを更新することで, 同一のデータベースが作成される.

図 2.1 に, ブロックチェーンの詳細な構造を示す. 各ブロックデータの完全性 (Integrity) は, 暗号学的ハッシュ関数を用いた固有のハッシュ値により, 保証される. 各ブロックは, 直前のブロックハッシュ値を含むことで, 直前のブロックに接続される. 赤で提示されているブロックは, 合意形成の過程で, 正しいチェーンに含まれると認められなかった孤立ブロック (Orphaned block) である.

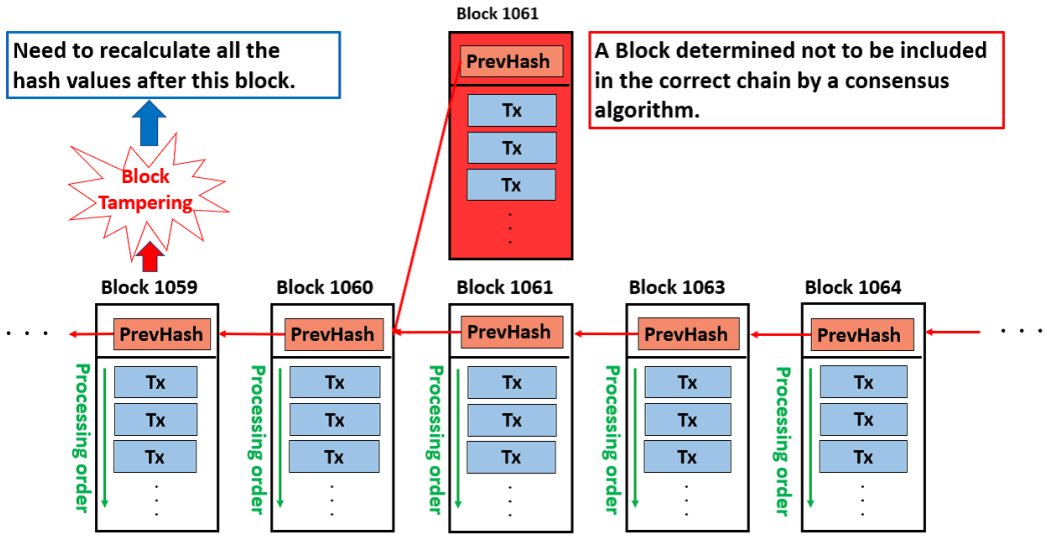


図 2.1: ブロックチェーンの構造. ブロック内のトランザクション処理順は明確に定まっている. 各ブロックは, 直前のブロックハッシュ値を含むことで, 連結される.

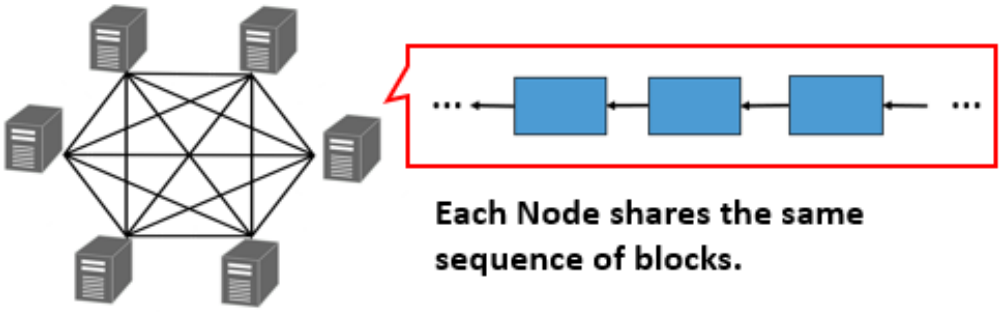


図 2.2: ブロックチェーンの P2P ネットワーク. 各分散ノードが, 合意によって正しいブロックチェーンを選択し, トランザクションを順番に処理してデータベースを更新

合意形成は, ブロックチェーンの P2P ネットワーク上で実行される (図. 2.2). 合意形成アルゴリズムの種類は, 2022 現在, Bitcoin や Ethereum で採用されている PoW (Proof of Work) [1, 31], Hyperledger プロジェクト等で採用される PBFT (Practical Byzantine Fault Tolerance) [32], Ethereum で今後採用が期待されている PoS (Proof of Stake) [33, 34], など多数存在する. これらアルゴリズムの実行方法, 各種パラメータの設定により, ブロックチェーンシステムの挙動は大きく変化する. 例えば, Bitcoin では, チェーン成長時間の間隔は平均 10 分になるようにフィードバック制御 [21] されており, Ethereum では平均約 14 秒程度になるように調整される [35].

2.2 ブロックチェーンシステムのトランザクション承認遅延

前節で説明したように, トランザクションが正しいチェーン内のブロックに取り込まれる事で, ブロックチェーンの分散データベースの更新は完了する. この手続きを, トランザクション承認と呼ぶ. ネットワークを介した合意形成を経た後に, ブロック単位でトランザクションを処理するため, 以下の二つの原因でトランザクション承認遅延が発生する.

- チェーン成長の振舞いによる, トランザクション承認時刻の制約
- ブロック内トランザクション数の限界による, キューイング処理

一点目は, トランザクションの承認時刻と, ブロックが正しいチェーンに含まれる時刻が厳密に一致するために, トランザクションを P2P ネットワークに送信してから承認されるまでに遅延が発生することを意味する. ここで, 一秒当たりに正しいチェーンの成長する長さをチェーン成長速度 v_{cg} と定義する. 仮に, ある時刻にネットワークに送信されたトランザクションが, 正しいチェーンを伸ばす直近のブロックに必ず含まれる場合, 承認遅延時間の期待値は平均チェーン成長時間 $T_{cg} = 1/v_{cg}$ 以下となる.

二点目は, 一つのブロックで処理可能な数を超過するトランザクションがシステム内に流入した場合に, キューイング処理によるトランザクション承認遅延が生じることを意味

する. ブロック内の最大トランザクション数 B はブロックサイズと呼ばれ, ネットワーク上のノードに対する, 効率的なブロック伝搬を可能にするために設定される. トランザクションの到着率 λ が, 次式を満たすとシステムは安定でなくなり, 承認遅延時間が急速に増加することになる.

$$\lambda > \frac{B}{T_{cg}}. \tag{2.1}$$

上記二つの原因によるトランザクション承認遅延は, 同時に改善することができない. 仮に, チェーン成長速度 v_{cg} を増加させて平均チェーン成長時間 T_{cg} を減少させる場合, ネットワーク上のノードにブロックを高速で受信させ合意形成を行わせるために, ブロックサイズ B を減少させる必要がある (その逆もまた然りである). つまり, 大量のトランザクションがシステムに流入する (式 2.1 が満たされる) 場合, トランザクションを短時間で処理できなくなる (トランザクション承認遅延が増加する). このことは, ブロックチェーンのスケーラビリティ問題 [10, 11] と知られている.

スケーラビリティ問題を解決するために, ブロックチェーン外部 (オフチェーン) のデータベースを利用するオフチェーン技術が提案されている. オフチェーン技術は, 土台となるブロックチェーンの分散データベース, 暗号技術, 新たに導入されるオフチェーンプロトコルにより, オフチェーン上のデータの完全性 (Data integrity) を保証する. この結果, ブロックチェーンストレージの更新回数や利用容量が減ることで, トランザクションの送信回数やデータサイズが減れば, ブロックサイズ B を実質的に増加させたことになる. つまり, キューイング処理由来のトランザクション承認遅延が改善される. 次節では, オフチェーン技術の要素技術として頻繁に利用されるステートツリーについて解説する.

2.3 ステートツリーを用いたデータの完全性保証

本節では, データの完全性を証明可能にするステートツリー [3, 36, 37] について説明する. トランザクション承認遅延を改善するために提案されているオフチェーン技術は,

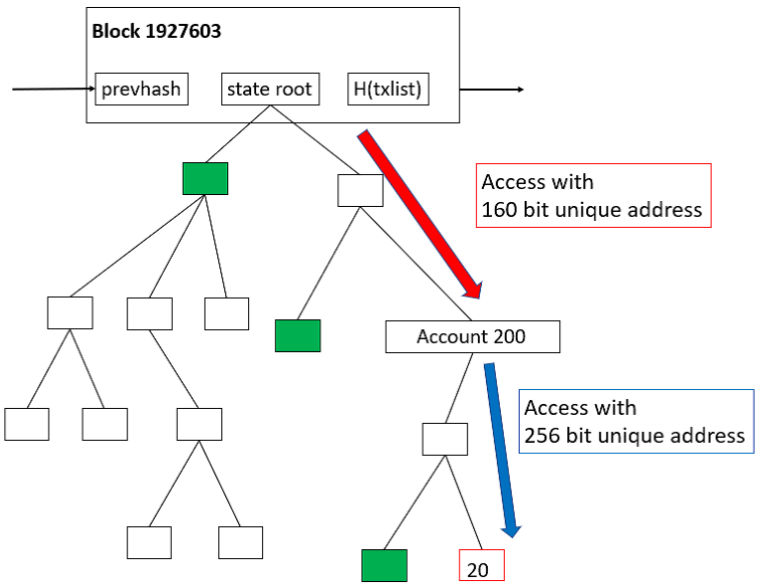


図 2.3: Ethereum が採用するデータ完全性保証用のステートツリー

ブロックチェーン外部にあるデータの完全性を保証する．このため，ステートツリーは，オフチェーン技術の要素技術として頻繁に採用されている．

Ethereum [3] は，ブロックチェーンのデータベース (オンチェーンストレージ) に任意のデータを保存可能であり，そのデータの完全性をステートツリーにより保証している．任意データは，コントラクトと呼ばれる単位で管理される Key-Value ストアにデータブロックと呼ばれる単位で格納される．ステートツリーはハッシュツリーの一種であり，そのリーフノードは対応するデータブロックに暗号学的ハッシュ関数を施した値を持つ．ステートツリーのルート (ステートルート) は，ブロックチェーン上のブロック内に保存されており，リーフノードと関連する中間ノードを用いたマークル証明 [38] を実行することで，対象のデータブロックの完全性が保証される．

図 2.3 に，Ethereum で用いられているステートツリーを示す ([36] を参考に作図)．Ethereum 上のデータベースは，160 bit のユニークな ID を持つアカウント (これはユーザーかコントラクトのどちらかである) と呼ばれる単位で管理され，アカウントがコントラクトならば 256 bit の Key-Value ストアを所有している．つまり，Ethereum では，コ

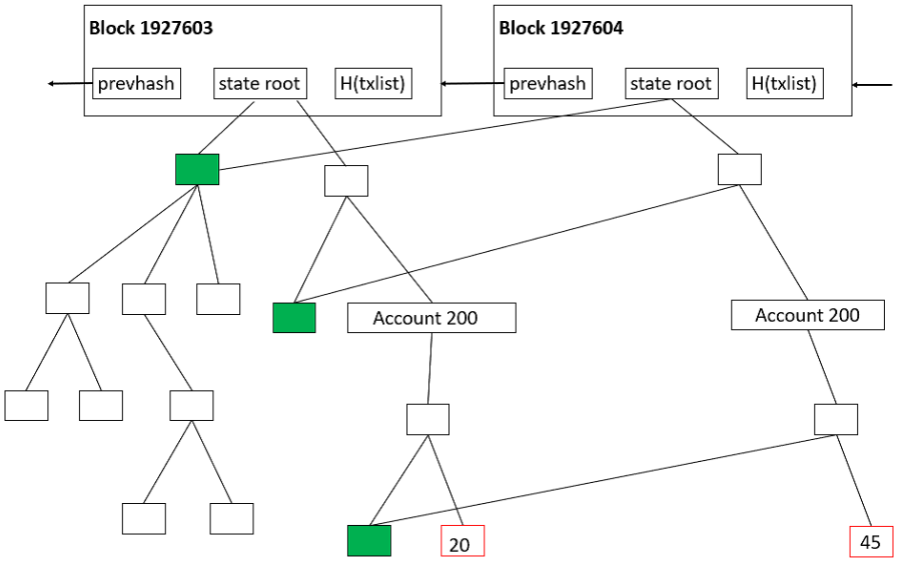


図 2.4: Ethereum のステートルートが更新される様子

ントラクトの持つ 160 bit の ID と, Key-Value ストアのキーである 256 bit, 合計 416 bit を用いて, 任意のデータを格納したデータブロックへアクセス可能である. 図 2.3 内の赤色で囲われた 20 という値を保存しているデータブロックは, 緑色の中間ノードのハッシュを用いたマークル証明により, 完全性が保証される.

図 2.4 は, ステートルートがどのように更新されるかを示した概念図である ([36] を参考に作図). ステートツリー内のデータブロックが, トランザクションにより更新された場合, ステートルートの再計算は対数オーダーで実施できる. データブロックにより更新される中間ノードの値のみを変更し, ステートルートを再計算可能なためである (図 2.4 内の緑色の中間ノードのハッシュ再計算は不要である). このステートルート更新手続きを, ブロック内の全トランザクションに対して実行することで, 新しいブロックのステートルートが計算される.

ステートツリーは, オフチェーン技術を導入したシステムである Arbitrum [15] や TrueBit [39] で採用されている. これらシステムの動作原理については, 次節で説明する.

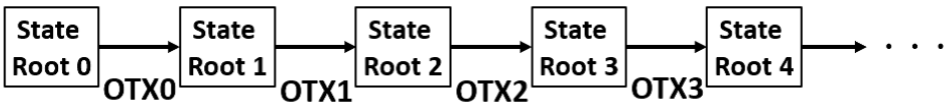


図 2.5: オフチェーントランザクションが処理される度に更新されるステートルート群

2.4 ステートツリーを用いたオフチェーン技術である ORU

オフチェーン技術は, 土台となるブロックチェーンの分散データベース, 暗号技術, 新たに導入するオフチェーンプロトコルにより, ブロックチェーンの外側 (オフチェーン) のデータの完全性を保証可能にする. 節 2.2 で説明したように, オフチェーン技術は, トランザクションの送信回数やデータサイズを減少させることで, ブロックサイズ B を実質的に増加させる手法として着目されている.

Arbitrum [15] や TrueBit [39] は, オフチェーン技術として, ステートツリーに基づく証明メカニズムを用いた ORU (Optimistic RollUp) を採用している. ORU は, オフチェーン上の任意のデータの完全性を保証することができ, 複雑な暗号技術を必要としない点もあり, メジャーなオフチェーン技術である. 本節では, ORU の動作メカニズムについて説明する.

ORU で用いられるステートツリーは, ハッシュツリーであることは前節と変わらないが, 各リーフノードにオフチェーン上のデータブロックに関する暗号学的ハッシュが付与されている. このステートツリーのルートハッシュは, 前節と同様にステートルートと呼ばれている. ORU では, ステートルートをブロックチェーン上 (オンチェーン) のストレージに一定周期で保存している. 例えば, Ethereum の場合, 図 2.3 内リーフノードのデータブロックにステートルートを保存する.

図 2.5 に, ORU で計算されるステートルートを更新順に時系列に並べた様子を示す. 図 2.5 は, オフチェーン上のデータベースを更新するトランザクション (OTX) が処理される度に更新されるステートルート群の様子を示している. ブロックチェーン上に保存されたステートルートを用いたマークル証明により, オフチェーンデータの完全性を証明

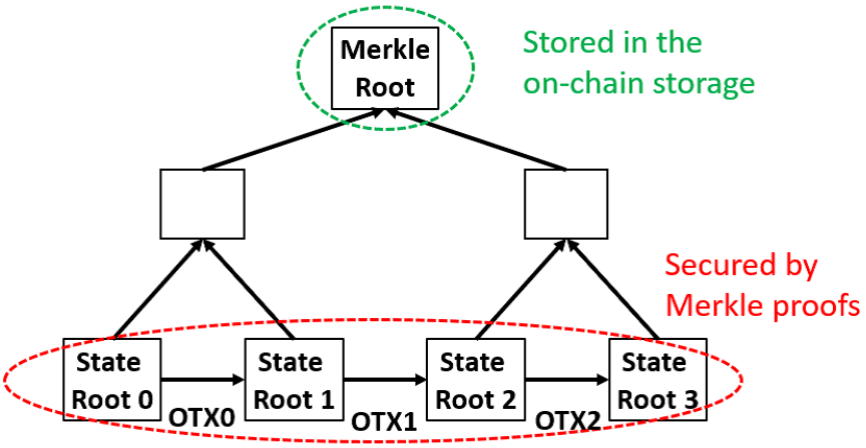


図 2.6: 各ステートルートの完全性を, オンチェーン上のマークルルートにより保証

するには, 全てのステートルートが OTX に記述されたルールに従って正しく更新されていなければならない.

ブロックチェーンの通常のトランザクション処理の正しさは, 合意形成により保証されている. しかし, 合意形成では, ブロックチェーン外部のデータを利用する OTX の処理の正しさを検証できない. このため, ORU では, 紛争解決プロトコルと呼ばれる新たな仕組みを導入することで, 更新されたステートルートの正しさを保証している.

2.4.1 紛争解決プロトコル

ORU で用いられる紛争解決プロトコルの仕組みを以下に示す. 最初に, 時系列上に並べられたステートルート群の完全性を保証するために, ステートルート群をリーフの入力としたマークルツリーのマークルルートをブロックチェーン上に登録する (図 2.6). このマークルルート登録者をオフチェーン管理者と呼ぶ. しかし, この段階では, 与えられたステートルート群の完全性は保証されるが, OTX を正しく処理した結果得られたステートルート群であるかどうか判定できない. この問題に対処するため, N_a 人の監査者に, 同じステートルート群とマークルルートを再計算させる. 仮に, オフチェーン管理者

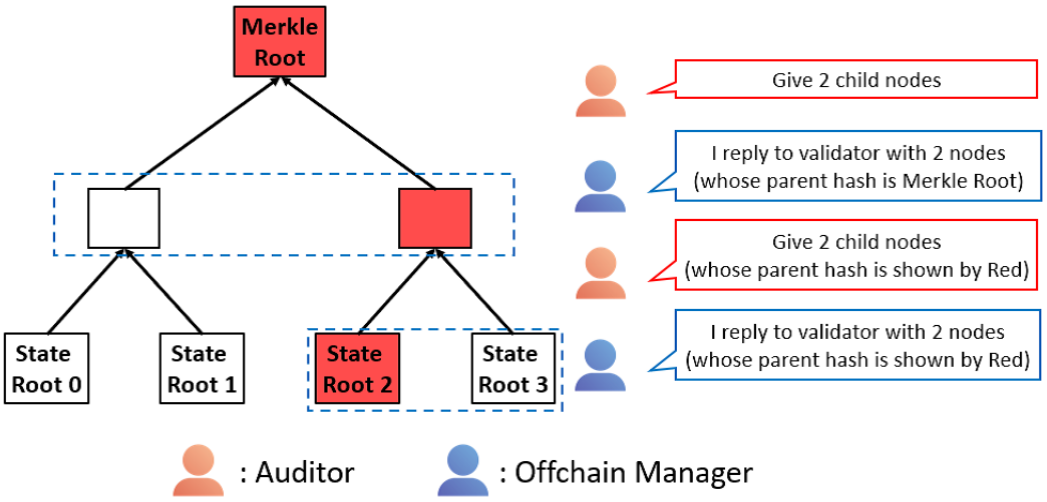


図 2.7: 対話型プロトコルにより, 異なっているステートルートを検出. 赤色で示されるノードのハッシュ値が, 監査者とオフチェーン管理者の間で異なっている.

が OTX を正しく処理していない場合, 監査人により再計算されたマークルルートは, ブロックチェーン上に登録されている値と一致しなくなる.

次に, OTX が初めて正しく処理されなかったステートルートを特定するため, 対話型プロトコルを実施する. 図 2.7 にその流れを示す. マークルルートの値が一致しない場合, 監査者は, 管理者に対して, マークルルートの二つの子ノードのハッシュ値を公開するように通知する. その後, 監査者は管理者の応答内容を見て, 自身の再計算したハッシュと異なっている子ノードをルートとした部分木について, 再帰的に同じ手続きをして探索していく. この際, 監査者は, 異なったハッシュ値を持つ最も左のノードを常に選択する.

以上の手続きで, 初めて正しく計算されなかったステートルートが, ブロックチェーン上に公開される. また, 最後に正しく計算されたステートルートは, 対話型プロトコルの途中で得られた中間ノードを用いたマークル証明により, 完全性を保証できることに注意する. 最後に, OTX が初めて正しく処理されなった隣接する二つのステートルートを, 監査者がブロックチェーン上に公開する. この結果, 一つの OTX による処理をオンチェーン上でエミュレートをすることで, オフチェーン管理者により完全性が保証されたオフ

チェーンデータベースを棄却すべきか判断可能になる (エミュレートに必要なオフチェーンデータは, 完全性を保証するマークル証明と共に, 監査者によってブロックチェーン上に保存される.). この判断結果は, 一人でも誠実な監査者がいれば常に正しいものとなる.

マークルツリーの木の高さを H とした場合, 一回の紛争解決プロトコルは, $2H+1$ 回のトランザクション送信が必要である. つまり, 紛争解決プロトコルは最悪の場合, オフチェーン管理者のマークルルート登録手続きを加えて, $N_a(2H+1)+1 = 2N_aH+N_a+1$ 回トランザクションを送信する手続きが必要となる (最善の場合, 紛争解決プロトコルが行われないため, 1 回のトランザクションで済む). このため, 不正を行うインセンティブを減らすため, オフチェーン管理者, 監査者になるには一定額のデポジットをする仕組み [34] を採用するシステムが多い. 対話型プロトコルの途中放棄, 紛争解決プロトコルの結果で主張が間違っている場合に, デポジットは破壊される.

第3章　関連研究

ブロックチェーンシステムにおいて, トランザクションは, 正規ブロックに含まれて初めて承認される. Bitcoin をはじめとしたブロックチェーンは, システムに流入するトランザクションが増加した場合に, トランザクション承認遅延が大きく増加するスケーラビリティ問題を抱えている. 本章では, 承認遅延の原因であるチェーン成長の振舞いとキューイング処理に関連した研究を紹介する.

3.1　チェーン成長の振舞いに関連する研究

3.1.1　平均チェーン成長時間の下限 [11]

[11] では, ネットワークのブロック伝搬性能から, 平均チェーン成長時間 (Chain growth time interval) T_{cg} の下限について分析している. 論文では, ブロックのデータサイズ block data size, そのブロックをブロックチェーンネットワーク上の $X\%$ のノードに伝搬するのにかかる時間 $X\%$ block propagation delay を用いて, $X\%$ effective throughput を次式で定義している.

$$X\% \text{ effective throughput} = \frac{\text{block data size}}{X\% \text{ block propagation delay}}. \tag{3.1}$$

$X\%$ effective throughput は, データサイズ block data size のブロックをネットワークに送信した場合に, $X\%$ のノードのブロック受信によって, トランザクションが承認されたと見做す場合の実行スループットと解釈できる. $X\%$ 実行スループットを用いて, 平均チェーン成長時間 T_{cg} の下限を次式で定義できる.

$$\frac{\text{block data size}}{X\% \text{ effective throughput}} < T_{cg}. \tag{3.2}$$

論文では, 2014 年 11 月から 2015 年 11 月までの Bitcoin に関するケーススタディを行っており, 50% effective throughput = 496 Kbps, 90% effective throughput = 55 Kbps という値を, ネットワーク上に配備したノードを用いて計測している. Bitcoin のブロックデータサイズの上限は 1 [MB] である [21]. block data size = 1[MB] とすると, 式 3.2 の左辺は, 50% effective throughput の場合は 16.1 [sec], 90% effective throughput の場合は 145.5 [sec] となり, Bitcoin の平均チェーン成長時間である 600 [sec] を共に下回ることが確認できる.

3.1.2 PoW ブロックチェーンのチェーン成長の振舞いに関連する研究

PoW ブロックチェーンでは, 合意形成アルゴリズムとして PoW (Proof of Work) を用いる. PoW では, 合意形成に参加するノード (マイナーと呼ばれる) が, 計算リソースを消費して暗号学的な問題を解くこと (マイニング) に成功した場合に, ブロックチェーンにブロックが接続されて成長する. 合意形成で選択されるチェーンは, 最も投入された計算リソースの多い最長チェーンである. このため, マイナーは常に最長チェーンを延ばそうとする. 問題の難易度は, 目標とする平均チェーン成長時間の理論値と, タイムスタンプから計算された平均チェーン成長時間の差を減らすように, フィードバック制御されている. 例えば, Bitcoin では, 平均チェーン成長時間 $T_{cg} = 600$[sec] になるよう, 2,016 ブロック毎に難易度調整している [21].

また, 前節で述べたようにブロックチェーンの P2P ネットワークには, ブロック伝搬遅延が存在する. このため, ある時刻にマイニングされたブロックは瞬時にネットワーク全体に伝わらない. 本節では, ブロック伝搬遅延が PoW ブロックチェーンのチェーン成長振舞いに与える影響を分析に関連する研究を紹介する.

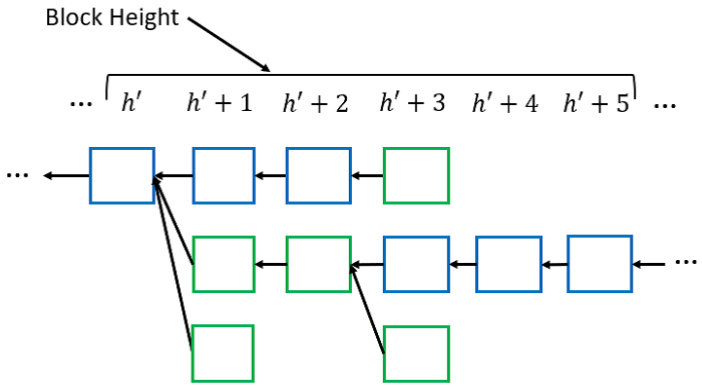


図 3.1: ブロック高が h' から $h'+5$ であるブロック群.

(1) PoW ブロックチェーンのフォークに関連する研究 [40]

C. Decker らは, 論文 [40] 内で, PoW ブロックチェーンのフォークについて分析している. P2P ネットワークにはブロック伝搬遅延が存在するため, 各マイナーにとっての最長チェーンは, ブロック受信状況によって異なるものとなる. このため, ほぼ同時刻に二つのブロックがマイニングされた場合, チェーンが二つ以上に分岐する可能性がある. この事象をフォークと呼ぶ.

C. Decker らによるブロックチェーンのフォークの定義を以下に示す. 図 3.1 は, ブロックチェーンがフォークしている様子である (本図は, 本論文著者が説明のために作成). ブロック高 h において, 合意形成時の候補となっているチェーン数を $|\beta_h|$ とする. つまり, ブロック高 h でフォークが発生する場合, $|\beta_h| > 1$ となる. 例えば, 図 3.1 の場合, $|\beta_{h'+1}| = 3 > 1$ となっている.

あるブロック高 h_s から h_e のフォーク確率 P_F は, その区間でフォークが発生しているブロック高の割合で定義される. 図 3.1 における, ブロック高 h' から $h'+5$ 間のフォーク確率は, $h'+1$, $h'+2$, $h'+3$ の場合に $|\beta_h| > 1$ であるため, $P_F = 0.5$ である. P_F の理論値は, 以下である ([40] 内の式 (2) を参照).

$$P_F = 1 - (1 - P_b(\alpha))^{\int_0^\infty (1-F(t))dt}. \tag{3.3}$$

高さ h で最初にマイニングされたブロックを $b_0(h)$ と定義する. 時間 t は, $b_0(h)$ をマイニングした時刻を 0 とした場合の経過時間である. $F(t)$ は, フォークが発生しない前提の下, 時刻 t に $b_0(h)$ を延長するのに使用される計算リソースの割合である. $P_b(\alpha)$ は, 1 秒以内に, 任意のマイナーにより新たなブロックがマイニングされる確率である[1]. 式 3.3 より, $F(t)$ が短時間で 1 に収束すると, フォーク発生確率 P_F が 0 に近づくことが理解される.

また, マイナーが自身の利益を最大化させるために, 意図的にフォークを発生させる利己的マイニング (Selfish Mining) という戦略も提案されている [41, 42]. 利己的マイニングでは, マイナーがある閾値を超える計算リソースを持つと, 常に最長チェーンを延ばすインセンティブが働かなくなる. この結果, フォーク発生確率が上昇する. 本研究では, ブロック伝搬遅延により発生するフォークに限定して取り扱い, 利己的マイニング由来のフォークは対象外とする.

(2) PoW ブロックチェーン成長の振舞い: 同期モデル

PoW ブロックチェーンの最長チェーン成長時間を分析するために, 二種類のモデルが提案されている. 同期モデルと非同期モデルである. 同期モデルでは, ブロック伝搬遅延が存在しないことを前提としている. 従って, 最長チェーン成長時間の分布は, ブロックを新たにマイニングするのにかかる時間分布 (マイニング成功時間の分布) と一致する.

C. Grunspan ら [31] は, マイニング成功時間が指数分布に従うと仮定 (この導出のレビューは, 本博士論文の著者が付録 A.1 に載せている) し, Bitcoin の二重支払い攻撃が成功する確率を閉形式で求めている. この式は, 離散モデルにおいて M. Rosenfeld [43] によって導出された式と一致するため妥当性がある.

[31] 内で, n ブロック分のマイニング成功時間 (つまり, 最長チェーンが n ブロック成長する時間) の累積分布関数 (CDF: Cumulative Distribution Function) である

[1] α は, マイニング成功の難易度を表すパラメータである. Bitcoin の場合, $\alpha = 1/600$ になるように難易度が調整される. α が大きいほど容易にマイニング可能である.

$G_{\mathrm{nd}}(n,t)$ $t(\geq 0)$ は, 次式で計算される.

$$G_{\mathrm{nd}}(n,t) = 1 - \exp(-\alpha t)\sum_{k=0}^{n-1}\frac{(\alpha t)^k}{k!}. \tag{3.4}$$

また, 確率密度関数 (PDF: Probability Density Function) $g_{\mathrm{nd}}(n,t)$ は, 次式で計算される.

$$\begin{aligned} g_{\mathrm{nd}}(n,t) &= \frac{d[G_{\mathrm{nd}}(n,t)]}{dt}, \\ &= \frac{\alpha^n}{(n-1)!}\cdot t^{n-1}\exp(-\alpha t). \end{aligned} \tag{3.5}$$

同期ネットワークでは, α はブロック生成速度であり, チェーン成長速度 v_{cg} でもある. つまり, 平均チェーン成長時間は $T_{cg} = 1/v_{cg} = 1/\alpha$ となる. $n = 1$ の時, 式 3.5 は以下のように書き直される.

$$g_{\mathrm{nd}}(t) = \alpha\exp(-\alpha t). \tag{3.6}$$

式 3.6 の右辺は, パラメータ α の指数分布の PDF である. 表記の簡潔性のため, $G_{\mathrm{nd}}(1,t) \equiv G_{\mathrm{nd}}(t)$, $g_{\mathrm{nd}}(1,t) \equiv g_{\mathrm{nd}}(t)$ と定義する.

同期ネットワークという仮定を置けば, PoW ブロックチェーンのセキュリティや, パフォーマンス, ブロックチェーンアプリケーションの特性を評価する際に, 式 3.4, 3.5, 3.6 を利用することができる. 例えば, 式 3.6 は, Y. Kawase ら [13] が Bitcoin の平均トランザクション承認時間を理論的に導出する際に使用されている. しかし, 実際のネットワークにはブロック伝搬遅延が存在する. 従って, 式 3.4, 3.5, 3.6 を利用するためには, ブロックチェーンネットワークが同期していると見做せるかどうか判断する必要がある.

3.1.3 PoW ブロックチェーン成長の振舞い: 非同期モデル

非同期モデルでは, P2P ネットワークにブロック伝搬遅延が存在する. ブロック伝搬遅延の影響を考慮するために, 有界遅延モデル (Bounded delay model) が提案されている [22, 23, 44]. 有界遅延モデルでは, ブロック伝搬遅延時間の上限 $D(\geq 0)$ を仮定する.

つまり, 最悪の場合でも, 各ブロックがマイニングされてから D 秒後には, 全ノードがブロックを受信する. 従って, 有界遅延モデルでは, ブロック伝搬遅延時間が常に一定値 D であるネットワークを考えることで, 最長チェーンが n ブロック成長するのにかかる時間の CDF の下限 $G_{\mathrm{cd}}(n,t)$ を計算可能である.

$$G_{\mathrm{cd}}(n,t) = \begin{cases} G_{\mathrm{nd}}(n, t-nD) & (t \geq nD), \\ 0 & (0 \leq t < nD). \end{cases} \tag{3.7}$$

また, PDF $g_{\mathrm{cd}}(n,t)$ は次式で与えられる.

$$g_{\mathrm{cd}}(n,t) = \begin{cases} g_{\mathrm{nd}}(n, t-nD) & (t \geq nD), \\ 0 & (0 \leq t < nD). \end{cases} \tag{3.8}$$

表記の簡潔性のため, $G_{\mathrm{cd}}(1,t) \equiv G_{\mathrm{cd}}(t)$, $g_{\mathrm{cd}}(1,t) \equiv g_{\mathrm{cd}}(t)$ と定義する.

また, ブロック伝搬が常に一定時間 D だけ遅れ, その間マイナーがマイニングを停止するネットワークを考えることで, 平均チェーン成長時間 T_{cg} の上限を次式で計算可能である (ここで, $\alpha = v_{cg}$ はブロック生成速度である).

$$T_{cg} \leq \int_0^\infty g_{cd}(t)dt = \frac{1}{\alpha} + D = \frac{1}{v_{cg}} + D \tag{3.9}$$

有界遅延モデルを利用するには, ネットワークのブロック伝搬遅延の上限 D を評価する必要がある. Y. Sompolinsky ら [22] は, Bitcoin ネットワークでの観測結果 [40] を用いて, ブロックのデータサイズに応じた伝搬遅延の上限を算出して D を決定している. しかし, この手法では, ブロックがマイニング時刻と, ブロック受信時刻に関する大量のデータが必要となり, 多数のノードをネットワーク上に配備してデータを測定しなくてはならない. R. Pass ら [23] は, ネットワークにおけるリンクの最小帯域とネットワークの直径 (Diameter) が予め与えられているという仮定で, 上限値 D を推定している. しかし, リンクの最小帯域評価には, 自身のコントロール下にない 2 つのノード間の計測データを必要とするため, 原理的に実測不可能である.

3.2 ブロックチェーンのキューイング処理に関連する研究

本節では, ブロックチェーンのキューイング処理に関連した研究を紹介する. 待ち行列理論に基づき, ブロックチェーントランザクション承認時間の評価を行っている. 一方, これら研究では, ブロックチェーンの外側を利用するオフチェーン技術に関するキューイング処理のモデル化は行っていない点に注意する.

3.2.1 Bitcoin の平均トランザクション承認時間評価 [13]

Y. Kawase ら [13] は, Bitcoin のトランザクション承認時間を評価するための待ち行列モデルを提案した. 提案された待ち行列モデルは, 以下のような性質を持つ. トランザクションは, レート λ のポアソン過程に従って到着する. トランザクションがシステムに到着すると, そのトランザクションはまず待ち行列に入る. 待ち行列内のトランザクションは, マイニング開始時に一つのブロックにグループ化される. 一つのブロックに含まれるトランザクション数の最小値は 0 であり, 最大値は B である. 仮に, マイニング中のブロック内トランザクション数が B より少ない場合でも, 新たに到着したトランザクションは常に後続のブロックに含まれる. マイニングに成功すると, グループ化されたトランザクションが承認され, 次のマイニングプロセスがすぐ開始される.

$S_i(i = 1, 2, ...)$ を i 番目ブロックのマイニング成功時刻とする (ここで, $S_0 = 0$ とする). j 番目ブロックのマイニング成功にかかる時間 $S_j - S_{j-1}(j = 1, 2, ...)$ は独立同分布であり, レート α の指数分布 $G_{\mathrm{nd}}(t)$ に従う確率変数 S となる. つまり, 同期モデルを仮定しており, $E[S] = T_{cg} = 1/v_{cg} = 1/\alpha$ となっている.

以上の前提から平均トランザクション承認時間の理論値 $E_{\mathrm{tx}}[T]$ を求めている. Bitcoin システムから得た二年分 (2013 年 10 月から 2015 年 9 月) の計測値 $B = 1750$, $E[S] = 1/\alpha = 544.1$, $\lambda = 0.9710$ を用いて理論値 $E_{\mathrm{tx}}[T] = 1112.0$[sec] を求めている. 実測値は 1127.2[sec] となっており, モデルの妥当性が確認されている.

3.2.2 ブロック作成処理を考慮したトランザクション承認時間評価 [24]

QL. Li ら [24] は, PoW ブロックチェーンに関して, ブロック構築 (Block-generation) と, マイニングプロセスを明確に分類した待ち行列モデルを提案した. 具体的には, Y. Kawase [13] のモデルに, 新たにブロック構築と呼ばれる手続きを設定し, 二段階のバッチ処理を行う待ち行列モデルを導入した. ブロック構築プロセスとは, マイニング対象のブロックについて, トランザクション処理や検証手続きを行うことである. Bitcoin であれば, ブロック内の全トランザクションの UTXO (Unspent transaction output) [21] の入出力が一致していることの確認や, ディジタル署名検証が該当する.

QL. Li らのモデルでは, ブロック構築とマイニングプロセス各々のサービス時間がパラメータ μ_1, μ_2 の指数分布に従うと仮定している. このため, より一般的な分布に対応したモデル確立が将来課題であると述べている.

第4章　合意形成アルゴリズム PoW のチェーン成長の振舞い分析

チェーン成長の時間分布を評価することは, ブロックチェーンアプリケーション開発に必須である. トランザクションは, 正しいチェーンに含まれて初めて処理が実行されたと見做されるためである.

本章では, コンセンサス (合意形成) アルゴリズム PoW (Proof of Work) に着目し, チェーン成長の振舞いを数学的に評価する予測モデルを与える. 本予測モデルを用いることで, 従来の評価手法で必要とされる「ブロック伝搬遅延」の上限という計測困難なパラメータを前提とせずに, 単独ノードで容易に観測可能な「フォーク確率」を用いてチェーン成長の時間分布を評価可能となる.

本章の内容は, 第一著者として発表した論文 [27] をベースとしている.

4.1　背景

合意形成アルゴリズム PoW は, 特定管理者を必要としない Bitcoin [1] の土台となるブロックチェーン実現のために提案された. PoW では, 合意形成参加ノード (マイナーと呼ばれる) が, 計算リソースを消費して暗号学的な問題を解くこと (マイニング) に成功した場合, ブロックチェーンにブロックが接続されて成長する. 合意形成により選択されるブロックチェーンは, 最も計算リソースを消費して作成されたと考えられる最長チェーンである. このため, マイナーは常に最長チェーンを成長させようと試みる. PoW は, 合意形成に計算リソースを要求するため, シビル攻撃 [45, 46] に耐性があることで大きな注目を集めている.

ブロックチェーンシステムでは, トランザクション承認時刻は, そのトランザクションを含んでいるブロックが最長チェーンに接続する時刻と厳密に一致する. 従って, 最長チェーンの成長時間分布は「トランザクション承認遅延」の要因であり, ブロックチェーンを利用するアプリケーションに大きな影響を与える. 例えば, 「トランザクション承認遅延」により, ブロックチェーンの分散データベースを参照結果が変わる可能性がある (他の例については, 節 を参照).

このような観点から, ブロックチェーン成長振舞いのモデル化し, 「トランザクション承認遅延時間」の予測モデル構築が求められている. PoW において, マイニングの成功は確率的な事象である. 従って, 最長チェーン成長に必要な時間は, ある確率分布に従う確率変数と見做せる. C. Grunspan ら [31] は, 最長チェーン成長時間が指数分布に従うモデルを提案した. Y. Kawase ら [13] は, 上記のモデルを採用し, Bitcoin トランザクション承認遅延時間の平均値を理論的に導出した. しかし, チェーン成長時間が指数分布に従うモデルは, P2P ネットワーク上にブロック伝搬遅延がないことを前提としている.

より現実的なシナリオを考慮するため, 「ブロック伝搬遅延」の上限を仮定した有界遅延モデル (Bounded delay model) [22, 23, 44] が提案されている. しかし, 本モデルを利用するには, ネットワーク上に多数のノードを配備して得られる多数のブロック受信時間データから, 「ブロック伝搬遅延」の上限を評価する必要がある. この試みは, [40] で行われているが, 2 か月間, 3,048 個のノードを Bitcoin ネットワークに接続しており, 実測コストが非常に高い.

また, PoW ブロックチェーンには, 合意形成の選択候補となるチェーンが二つに分岐する「フォーク」と呼ばれる現象が存在する. 「フォーク」の生起確率は, 「ブロック伝搬遅延時間」が増加するほど, 増加する [40]. フォーク現象は, マイニングに成功したあらゆるブロックを受信し続けることで, 単独ノードであっても容易に観測可能である.

本研究では, 容易に観測可能な「フォーク確率」を用いて, 最長チェーン成長時間の分布を評価する手法を提案する. 具体的には, 最長チェーン成長時間の累積分布関数 (CDF: Cumulative Distribution Function) の下限を閉形式で導出する. CDF の下限を用いて,

チェーン成長速度 v_{cg} の下限を得ることができる. また, CDF の下限の逆関数を導出し, 最長チェーンが n ブロック分成長する時間分布を, 逆関数法により評価する手法も提案する. 最後に, 導出した下限と有界遅延モデルに基づく下限を比較するため, ネットワークシミュレーションを行った. 数値計算例によって, ブロック生成速度 α, ブロックデータサイズが CDF の下限に与える影響を示した.

4.2 PoW ブロックチェーンのフォークについて

本研究では, 論文 [40] 内のフォーク定義 (3.1.2 内の (1) で説明) を採用する. つまり, あるブロック高に二つ以上のチェーン候補がある場合に, そのブロック高でフォークが発生していると定義する. また, あるブロック高 h_s から h_e のフォーク確率 P_F は, その区間でフォークが発生しているブロック高の割合で定義する.

4.2.1 新たなフォーク確率公式の導入

論文 [40] 内で式 3.3 に代入することで得られるフォーク確率 P_F ([40] 内の式 (3) で計算) は, ブロック生成速度 α が十分に小さいこと ($\exp(-\alpha) \approx 1-\alpha$ と近似可能) を前提としている.

そこで, 本研究では, 上記前提を必要としない式 4.1 を導入する (本式の導出は, 付録 A.2 に載せた).

$$P_F = 1 - \exp\big(-\alpha \int_0^{\infty} (1 - F(t))dt\big). \tag{4.1}$$

ある高さ h において, 最初にマイニングされたブロックを $b_0(h)$ とする. 時間 t は, $b_0(h)$ がマイニングされた時刻を 0 とした場合の経過時間である. $F(t)(0 < F(t) < 1)$ は, フォークが発生しない前提の下, 時刻 t にブロック $b_0(h)$ を延長する計算リソースの割合である. つまり, $F(t) = 1$ となる時刻 t に, 全マイナーにブロック $b_0(h)$ が伝搬されたことになる. $F(t)$ はブロック $b_0(h)$ の伝搬速度を表現するパラメータといえる. $t \approx 0$

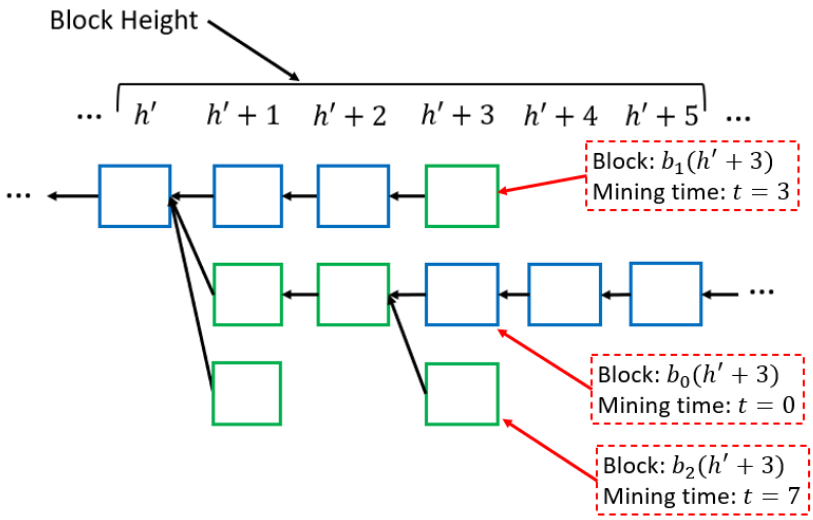


図 4.1: ブロック高 h' から $h'+5$ の間でマイニングされたブロック．青色のブロックは，あるブロック高で最初にマイニングされたブロックである．

で $F(t)=1$ となる場合，ブロック $b_0(h)$ はネットワーク全体に即伝搬されたことになり，式 4.1 のフォーク確率 $P_F=1$ となる．

図 4.1 は，ブロック高 h' から $h'+5$ の間にマイニングされたブロック群である．高さ $h'+3$ に着目すると，ブロック $b_0(h'+3)$ がマイニングされた時刻を $t=0$ とすると，$t=3$ にブロック $b_1(h'+3)$ がマイニングされ，$t=7$ にブロック $b_2(h'+3)$ がマイニングされている．このため，ブロック高 $h'+3$ において，フォークが発生している．また，ブロック高 $h'+1$, $h'+2$, $h'+3$ でフォークが発生しているため，高さ h' から $h'+5$ の間におけるフォーク確率は $P_F=0.5$ となる．

4.3 フォーク確率に基づく最長チェーン成長時間の分析

本節では，最長チェーン成長時間の CDF である $G(t)$ の閉形式の下限を導出する．また，ブロック生成速度 α とブロックデータサイズの変更が，提案した下限と，有界遅延モデルに基づく下限に与える影響を，シミュレーションを用いて明らかにする．

4.3.1 最長チェーン成長時間の CDF $G(t)$ の下限導出の流れ

本節では, 以下の手順で最長チェーン成長時間の CDF の下限を求める方法を説明する. まず, フォークが発生していない環境下で, 最長チェーンの延長に使用されている計算リソースの割合 $F(t)$ (4.2.1 を参照) の下限 $F_{\mathrm{lb}}(t)$ を導出する. 次に, $F(t)$ の代わりにその下限 $F_{\mathrm{lb}}(t)$ を, 最長チェーン成長時間の CDF $G(t)$ が満たすべき微分方程式に代入する. この微分方程式の解が, $G(t)$ の下限となる.

(1) フォーク確率を用いた $F(t)$ の下限導出

フォーク確率 P_F を用いて, $F(t)$ の下限 $F_{\mathrm{lb}}(t)$ を導出する. まず, 式 4.1 から, 加重平均遅延 (Weighted average delay) [23] T_w を以下のように計算する.

$$\begin{aligned} T_w &\equiv \int_0^\infty (1 - F(t))dt, \\ &= \frac{-\log(1 - P_F)}{\alpha}. \end{aligned} \tag{4.2}$$

式 4.4.2 は, $0 \leq 1 - P_F < 1$ であるため, フォーク確率 P_F が小さい程, T_w は小さくなる. つまり, P_F が小さい程, $F(t)$ は 1 に速く収束する. このことから, P_F を固定することで, 任意の時間 $t \geq 0$ における $F(t)$ の下限が与えられる. $F(\omega_\gamma) = \gamma$ となるような γ と ω_γ があるとすると, 式 4.4.2 から次の不等式が成り立つ.

$$\begin{aligned} T_w &= \int_0^\infty (1 - F(t))dt, \\ &\geq \int_0^{\omega_\gamma} (1 - F(t))dt, \\ &\geq \int_0^{\omega_\gamma} (1 - \gamma)dt = (1 - \gamma)\omega_\gamma. \end{aligned} \tag{4.3}$$

上の不等式は, 次の不等式に簡略化できる.

$$\omega_\gamma \leq \frac{T_w}{1 - \gamma}. \tag{4.4}$$

ここで, 任意の時刻 $t \geq T_w$ で, $F_{\mathrm{lb}}(t)$ を $F_{\mathrm{lb}}(T_w/(1-\gamma)) = \gamma$ と定義する. 以上より, 任意の $0 \leq \gamma < 1$ について, $F(\omega_\gamma) = \gamma$ かつ $F_{\mathrm{lb}}(T_w/(1-\gamma)) = \gamma$ であり,

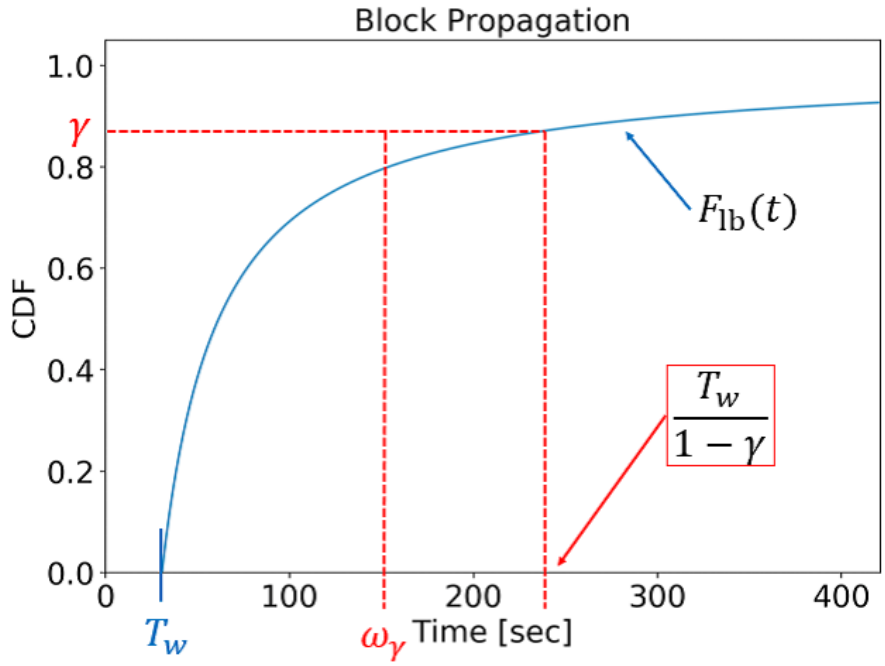


図 4.2: $F(t)$ の下限の例 $(\alpha = 1/600, P_F = 0.05)$

$\omega_\gamma \leq T_w/(1-\gamma)$ であるため, $F(t) \geq F_{\mathrm{lb}}(t)$ $(t \geq T_w)$ が成立する. また, $F_{\mathrm{lb}}(t) = 0$ $(0 \leq t < T_w)$ と定義することで, $F(t)$ の下限は次式により得られる.

$$F_{\mathrm{lb}}(t) = \begin{cases} 1 - \dfrac{T_w}{t} & (t \geq T_w), \\ 0 & (0 \leq t < T_w). \end{cases} \tag{4.5}$$

図 4.2 は不等式 4.4 と式 4.5 の関係を示したものである. 仮に, $F(t') < F_{\mathrm{lb}}(t')$ $(t' \geq 0)$ となるブロック伝搬が行われるケースを考えると, 分析のために与えられたフォーク確率 P_F より, フォーク生起確率が高くなる.

(2) 最長チェーン成長時間の CDF $G(t)$ の下限

最初に, 全マイナーが, 任意のブロック高 h において, 最初にマイニングされたブロック $b_0(h)$ を含むブロックチェーンのみを成長させるシナリオを考える (最長チェーンを成長させないマイナーは常に停止する). このシナリオにおける, 時刻 t の最長チェーン

成長速度 $v_{cg}(t)$ は, 厳密に $\alpha F\ (t)$ に等しくなる. 続いて, 上記シナリオの一部に変更を加え, $v_{cg}(t) = \alpha F_{\mathrm{lb}}(t)$ であると仮定する.

上記 2 つの仮定に従うシナリオは, 実際のケースと比較して, チェーン成長速度を悪化させるだけである. 従って, このシナリオにおける最長チェーン成長時間の CDF を考えれば, 実際の CDF $G(t)$ についての下限 $G_{\mathrm{lb}}(t)$ が得られる.

式 4.6 は, 短期間 $[t, t+\Delta t)$ の間にマイナーが最長チェーンを成長させることに成功する確率を意味する. 微分方程式 4.7 は, 式 4.6 の別表記である.

$$G_{\mathrm{lb}}(t+\Delta t) - G_{\mathrm{lb}}(t) = \Big(1 - G_{\mathrm{lb}}(t)\Big)\alpha F_{\mathrm{lb}}(t)\Delta t. \tag{4.6}$$

$$\therefore \quad \frac{\partial G_{\mathrm{lb}}(t)}{\partial t} = \Big(1 - G_{\mathrm{lb}}(t)\Big)\alpha F_{\mathrm{lb}}(t). \tag{4.7}$$

微分方程式 4.7 の解は, 式 4.8 で表される.

$$G_{\mathrm{lb}}(t) = \begin{cases} 1 - C\exp\Big(-\alpha\big(t - T_w \log t\big)\Big) & \big(T_w \le t < D\big), \\ 0 & \big(0 \le t < T_w\big). \end{cases} \tag{4.8}$$

$$\Big(C = \exp\big(\alpha \cdot T_w(1 - \log(T_w))\big)\Big)$$

定数値 C は, 初期条件 $G_{\mathrm{lb}}\big(T_w\big) = 0$ から得られる. 式 4.8 から計算される下限を図 4.3 に示す. 与えられたフォーク確率が小さい場合, 下限 $G_{\mathrm{lb}}(t)$ は同期モデルに基づく CDF (指数分布の CDF (式 3.6 を参照)) に漸近する. また, $G_{\mathrm{lb}}(t)$ を時間に関して微分して得られる関数を $g_{\mathrm{lb}}(t)$ と定義する. $g_{\mathrm{lb}}(t)$ は次式で計算できる.

$$g_{\mathrm{lb}}(t) = \begin{cases} C\exp\Big(-\alpha\big(t - T_w \cdot \log(t)\big)\Big) \times \alpha\Big(1 - \dfrac{T_w}{t}\Big) & \big(T_w \le t < D\big), \\ 0 & \big(0 \le t \le T_w\big). \end{cases} \tag{4.9}$$

$$\Big(C = \exp\big(\alpha \cdot T_w(1 - \log(T_w))\big)\Big)$$

図 4.4 の (a), (b) は, $g_{\mathrm{lb}}(t)$ を $\alpha = 1/600, 1/15$ の場合についてプロットしている. 以下では, $g_{\mathrm{lb}}(t)$ を用いて, 同期モデルに基づく指数分布と $g_{\mathrm{lb}}(t)$ のピアソン距離を計算することで, 同期モデルとの近似度を評価可能にする. また, (4) において, $g_{\mathrm{lb}}(t)$ を用いて, チェーン成長時間の平均値の上限を求める.

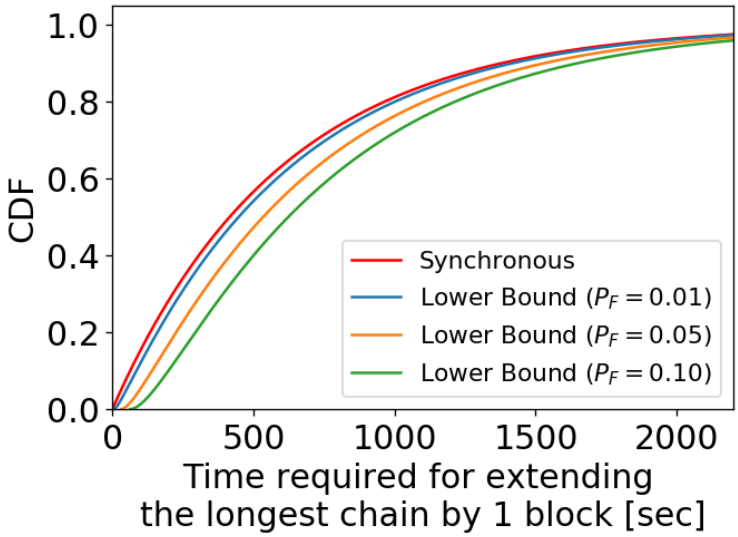


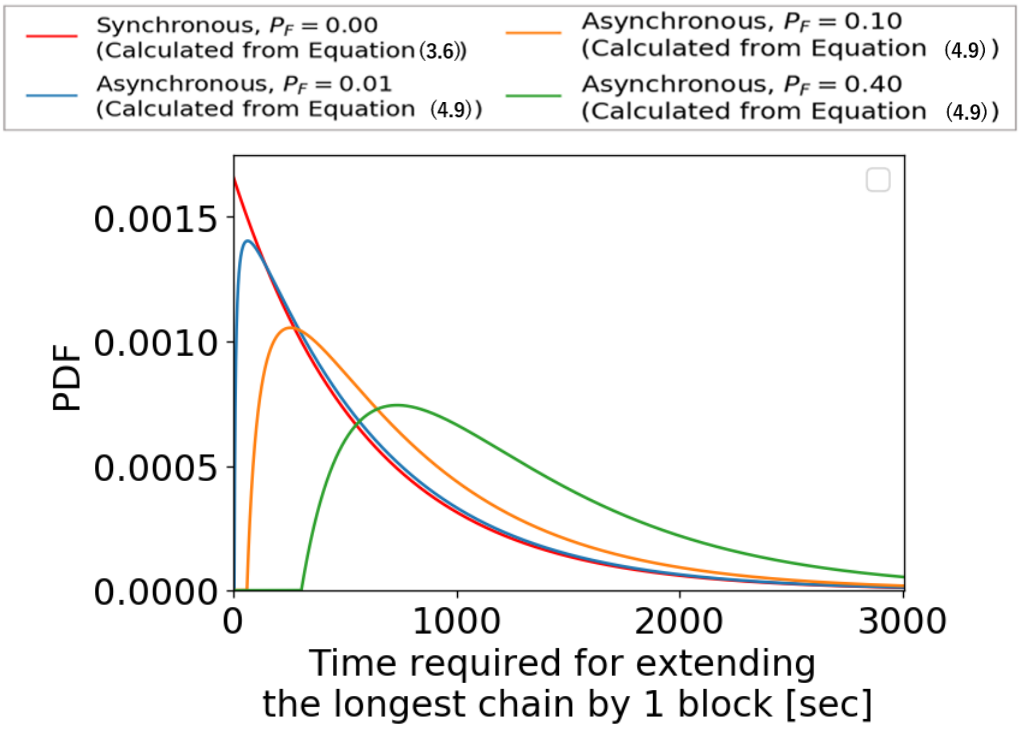
図 4.3: 式 4.8 から計算される最長チェーン成長時間の CDF の下限値 ($\alpha = 1/600$)

(3)　$g_{\mathrm{lb}}(t)$ と $g_{\mathrm{nd}}(t)$ を用いた同期モデルとの近似度評価

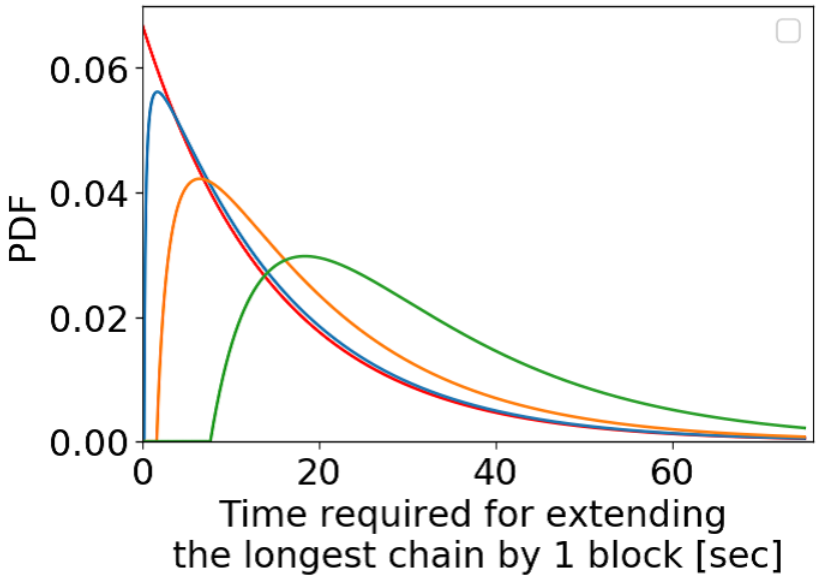
同期モデルとの近似度を評価するため, 観測された頻度分布が理論分布と同じかどうか判断する適合度検定に用いられる尺度, ピアソン距離 [47] を使用する. 近似度評価の際に, 最もよく使用される距離の一つである Kullback-Leibler 距離 [48] は, $t < T_w$ で $\log\left(g_{\mathrm{lb}}(t)/g_{\mathrm{nd}}(t)\right)$ が計算できないため採用しない. $g_{\mathrm{nd}}(t)$ と $g_{\mathrm{lb}}(t)$ とのピアソン距離は, 第二種不完全ガンマ関数 $\Gamma(a,x) = \int_x^\infty t^{a-1}dt$ を用いて, 式 4.10 より計算できる.

$$\int_0^\infty g_{\mathrm{nd}}(t)\left(\frac{g_{\mathrm{lb}}(t)}{g_{\mathrm{nd}}(t)} - 1\right)^2 dt = -1 + \exp(\alpha T_w) + (\alpha T_w)^{-2(\alpha T_w - 1)} \cdot \exp(2\alpha T_w) \cdot \Gamma(2\alpha T_w - 1, \alpha T_w), \quad (4.10)$$
$$\left(T_w = \frac{-\log(1-P_F)}{\alpha}\right).$$

式 4.10 の導出は付録 A.3 に載せる. 表 4.1 に, その距離を示す. ピアソン距離は, 2 標本検定や変化点検出 [49] により, ブロックチェーンネットワークがほぼ同期しているか否かを判断する指標として用いることができる. また, 導出したピアソン距離は観測されたフォーク確率にのみ依存するという特徴も持つ ($\alpha T_w = -\log(1-P_F)$ より).



(a) $\alpha = 1/600$



(b) $\alpha = 1/15$

図 4.4: 同期モデルに基づく $g_{\mathrm{nd}}(t)$ と導出した下限 $g_{\mathrm{lb}}(t)$ の比較

表 4.1: $g_{\mathrm{lb}}(t)$ と $g_{\mathrm{nd}}(t)$ のピアソン距離

Fork rate	The Pearson distance
0.01	0.020018
0.05	0.103501
0.10	0.219171
0.20	0.503638
0.40	1.453386

(4) 平均チェーン成長時間 T_{cg} の上限

$G(t)$ の下限 $G_{\mathrm{lb}}(t)$ を導出するシナリオでは, マイナーは各高さにおいて最も早く見つかったブロックのみを延長し, 最長チェーンの先頭ブロックが分からない場合はマイニングしない. 従って, このシナリオにおける最長チェーン成長時間の平均が, 通常のシナリオにおける平均チェーン成長時間 T_{cg} の上限となる. この上限は, 式 4.11 によって計算できる (式 4.11 の導出は付録 A.4 に掲載)

$$\int_0^\infty t \cdot g_{\mathrm{lb}}(t)\, dt = C \cdot \alpha^{-1-\alpha T_w} \Big(\Gamma\big(\alpha T_w + 2, \alpha T_w\big) - \alpha T_w \Gamma\big(\alpha T_w + 1, \alpha T_w\big) \Big) \quad (4.11)$$
$$\left(T_w = \frac{-\log(1-P_F)}{\alpha}\ ,\ C = \exp\big(\alpha \cdot T_w(1 - \log\ T_w)\big) \right)$$

計算結果を表 4.2 に示す. フォーク確率が低い程, チェーン成長時間の平均の上限が, 同期モデルのチェーン成長時間の平均に近づいていることが分かる.

(5) 最長チェーンが $n(>1)$ ブロック分だけ成長するのにかかる時間の CDF の下限

式 4.8 から, n ブロック分の最長チェーン成長時間の CDF の下限 $G_{\mathrm{lb}}(n,t)$ を閉じた形で導出するのは困難である. そこで, 逆関数法を用いたモンテカルロシミュレーションにより $G_{\mathrm{lb}}(n,t)$ を数値的に計算する. 逆関数法では, $G_{\mathrm{lb}}(t)$ の逆関数 $G_{\mathrm{lb}}^{-1}(u)$ を用いて,

表 4.2: 最長チェーンの平均成長時間の上限 ($\alpha = 1/600$)

Fork rate	An upper bound ($\alpha = 1/600$)	An upper bound ($\alpha = 1/15$)
0.0010	604.42 [sec]	15.11 [sec]
0.0025	609.71 [sec]	15.24 [sec]
0.0050	617.47 [sec]	15.44 [sec]
0.0100	631.15 [sec]	15.78 [sec]
0.0250	666.15 [sec]	16.65 [sec]
0.0500	716.33 [sec]	17.91 [sec]
0.1000	805.19 [sec]	20.13 [sec]
0.2000	968.91 [sec]	24.22 [sec]
0.4000	1305.18 [sec]	32.63 [sec]

$G_{\mathrm{lb}}(t)$ に従う確率変数を生成する. $G_{\mathrm{lb}}(t)$ の逆関数は, 式 4.12 のようにして計算できる.

$$G_{\mathrm{lb}}^{-1}(u) = -T_w \times W\Bigg(-\frac{\big((1-u)/C\big)^{1/(\alpha \cdot T_w)}}{T_w}\Bigg), \qquad (4.12)$$
$$\Big(C = \exp\big(\alpha \cdot T_w(1 - \log(T_w))\big)\Big).$$

ここで, $W(\cdot)$ は Lambert の W 関数である. $G_{\mathrm{lb}}^{-1}(u)$ に一様乱数 $u \in [0, 1]$ を代入すると, 与えられた分布に従う乱数が生成できる. n 個の確率変数の和を繰り返し生成することで, 図 4.5 の累積相対度数グラフが得られた.

4.3.2 ネットワークシミュレーションとその結果

最長チェーン成長時間に関する CDF の下限 $G_{lb}(t)$ を評価するために, Bitcoin を想定し, 最長チェーンの成長振舞いをシミュレートした. このシミュレーションでは, ブロック伝搬ノードとマイニング実行ノードの 2 種類のノードが存在する. Bitcoin の実装に従い, ブロック伝搬ノードは 8 つのアウトバウンド接続を維持し, すべてのインバウン

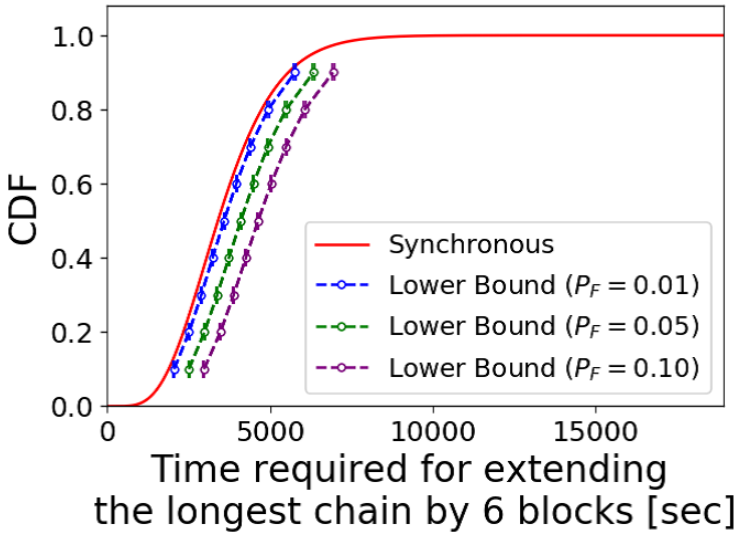


図 4.5: 数値計算により得られた最長チェーンが 6 ブロック成長するのにかかる時間の CDF の下限 (99.9% 信頼区間)

ド接続を受け入れる. マイニング実行ノードは, ランダムに 100 のブロック伝搬ノードに接続する (2014 年の Bitcoin ネットワーク上の最高度数のノードは 90 度以上である [50]). シミュレーションでは, ブロック伝搬ノード数を 5,000 とし, マイニング実行ノード数は 20 とした. 各リンクの伝搬遅延は, 正規分布 $\mathcal{N}(\mu, \sigma), \mu = \sigma = 100$[milliseconds] に従い, 各リンクの帯域は正規分布 $\mathcal{N}(\mu, \sigma), \mu = 4.0, \sigma = 0.8$[Mbps] に従う. いずれも負の値があれば再サンプリングした. ネットワークシミュレーションは 4 パターン実行した ($\alpha = 1/600$, ブロックサイズが 1 MB と 10MB, $\alpha = 1/15$, ブロックサイズは 25kB と 250kB). 各シミュレーションは, ブロック高が 50,000 に到達するまで実行された. 各シミュレーション結果を図 4.6, 4.7, 4.8, 4.9 に示す (表 4.3, 4.4 は図 4.6, 4.7 のシミュレーション結果の代表値を示している. 観測されたフォーク確率を用いて, 式 4.8 に基づき下限を計算する. また, 有界遅延モデルにより与えられる下限 $G_{\mathrm{cd}}(t)$ (式 3.7) を計算するために, シミュレーション実行中に, ブロックが全てのマイナーに伝搬されるまでの最長時間を有界遅延モデルの 上限 D として採用した.

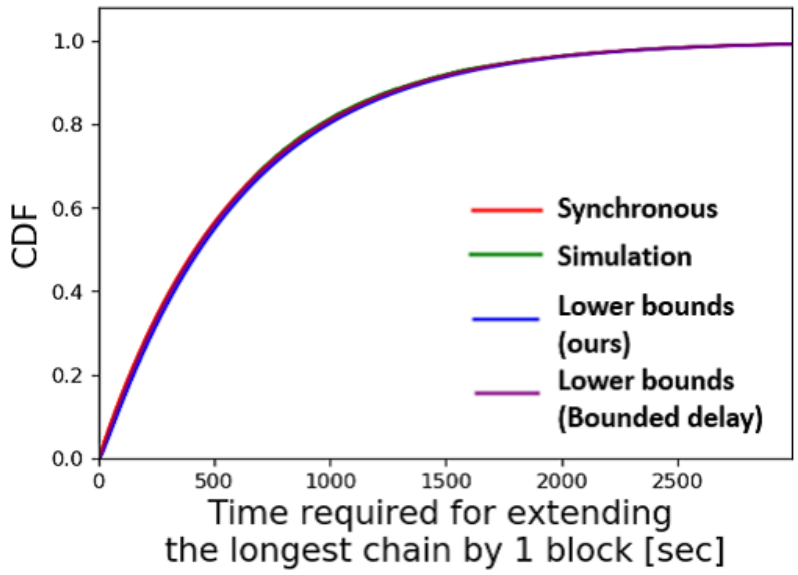


図 4.6: $\alpha = 1/600$, ブロックサイズ: 1 MB, 観測したフォーク確率 $\hat{P}_F$: 0.00636, 観測したブロック伝搬遅延の上限 $\hat{D}$: 7.1626 [sec]

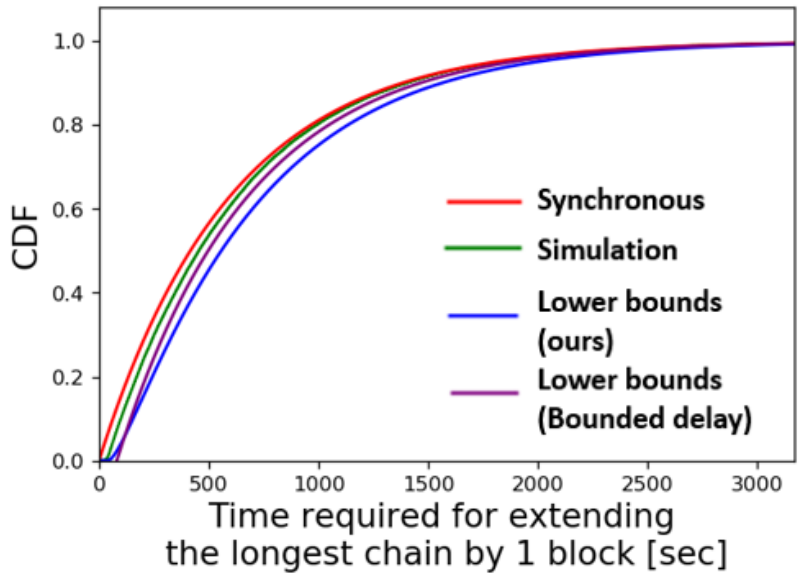


図 4.7: $\alpha = 1/600$, ブロックサイズ: 10 MB, 観測したフォーク確率 $\hat{P}_F$: 0.06182, 観測したブロック伝搬遅延の上限 $\hat{D}$: 81.0510 [sec]

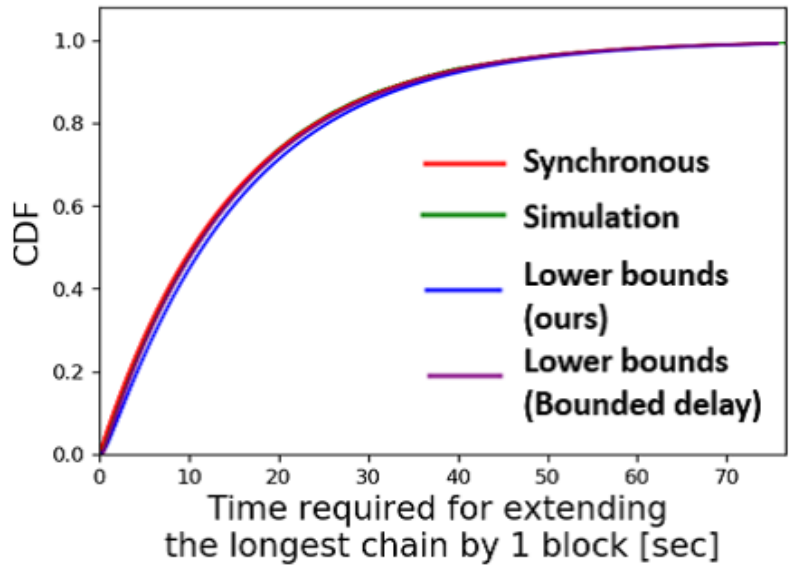


図 4.8: $\alpha = 1/15$, ブロックサイズ: 25 kB, 観測したフォーク確率 $\hat{P}_F$: 0.01554, 観測したブロック伝搬遅延の上限 $\hat{D}$: 0.4242 [sec]

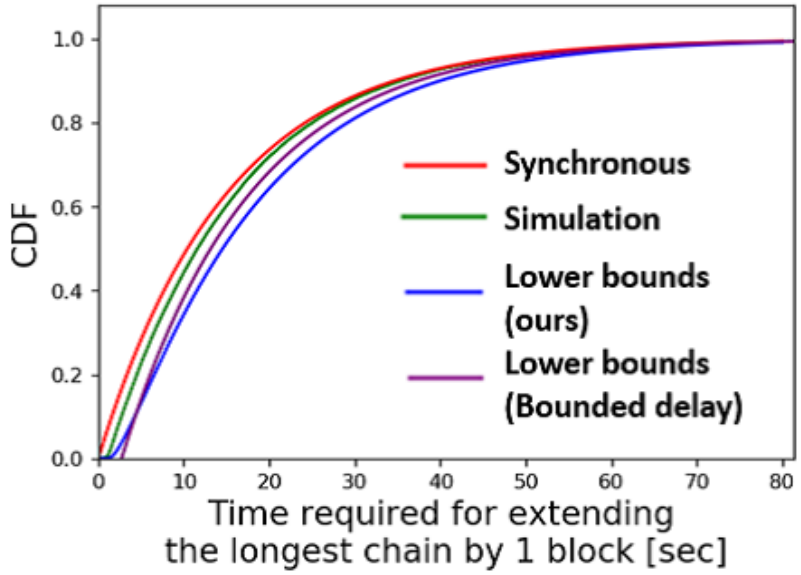


図 4.9: $\alpha = 1/15$, ブロックサイズ: 250 kB, 観測したフォーク確率 $\hat{P}_F$: 0.07522, 観測したブロック伝搬遅延の上限 $\hat{D}$: 2.7291 [sec]

表 4.3: $\alpha = 1/600, \hat{D} = 7.1626$, $\hat{P}_F = 0.00636$, 図 4.6 のシミュレーション値を利用.

Cumulative Probability	**Time** t_1 s.t. $G_{\mathrm{lb}}(t_1) = p$	**Time** t_2 s.t. $G_{\mathrm{cd}}(t_2) = p$
$p = 0.01$	15.1 [sec]	13.2 [sec]
$p = 0.05$	43.9 [sec]	37.9 [sec]
$p = 0.50$	437.9 [sec]	423.1 [sec]
$p = 0.95$	1824.9 [sec]	1804.6 [sec]
$p = 0.99$	2792.2 [sec]	2770.3 [sec]

表 4.4: $\alpha = 1/600, \hat{D} = 81.0510$, $\hat{P}_F = 0.06182$, 図 4.7 のシミュレーション値を利用.

Cumulative Probability	**Time** t_1 s.t. $G_{\mathrm{lb}}(t_1) = p$	**Time** t_2 s.t. $G_{\mathrm{cd}}(t_2) = p$
$p = 0.01$	64.0 [sec]	87.1 [sec]
$p = 0.05$	109.2 [sec]	111.8 [sec]
$p = 0.50$	556.7 [sec]	496.9 [sec]
$p = 0.95$	1986.9 [sec]	1878.5 [sec]
$p = 0.99$	2968.0 [sec]	2844.2 [sec]

4.4 提案手法と有界遅延モデルの組み合わせ

本節では, 提案したフォーク確率による評価手法と, 従来の有界遅延モデル評価手法の組み合わせを用いて, 平均チェーン成長時間 T_{cg} の上限と下限を評価する.

4.4.1 T_{cg} の上限の導出

有界遅延モデルには, ブロック伝搬遅延時間の上限 D が存在する ($D > T_w$ に注意する). そのため, 有界遅延モデルにおける $F(t)$ の下限 $F_{\mathrm{lb,D}}(t)$ を次式で表せる.

$$F_{\mathrm{lb,D}}(t) = \begin{cases} 0 & \big(t < D\big), \\ 1 & \big(t \geq D\big). \end{cases} \tag{4.13}$$

式 4.5 と式 4.13 の両方を満たす $F(t)$ の下限 $F_{\mathrm{lb,FD}}(t)$ を次式で定義する.

$$F_{\mathrm{lb,FD}}(t) = \begin{cases} 0 & \big(0 \leq t < T_w\big), \\ 1 - \dfrac{T_w}{t} & \big(T_w \leq t < D\big), \\ 1 & \big(t \geq D\big). \end{cases} \tag{4.14}$$

微分方程式 4.7 について, $F_{\mathrm{lb}}(t)$ の代わりに $F_{\mathrm{lb,FD}}(t)$ を代入して解くことで, フォーク確率と有界遅延モデルを考慮した場合の $G(t)$ の下限 $G_{\mathrm{lb,FD}}(t)$ が次式で与えられる.

$$G_{\mathrm{lb,FD}}(t) = \begin{cases} 0 & \big(0 \leq t < T_w\big), \\ 1 - C\exp\Big(-\alpha\big(t - T_w\log t\big)\Big) & \big(T_w \leq t < D\big), \\ 1 - C_1\exp(-\alpha t) & \big(D \leq t\big). \end{cases} \tag{4.15}$$

$$\Big(C = \exp\big(\alpha \cdot T_w(1 - \log(T_w))\big), \quad C_1 = \exp(\alpha D) \cdot C \cdot \exp\Big(-\alpha\big(D - T_w\log D\big)\Big)\Big)$$

定数値 C, C_1 は, $G_{\mathrm{lb,FD}}(T_w) = 0$ 及び, $t = D$ における $G_{\mathrm{lb,FD}}(t)$ の右側極限と左側極限が一致することから計算可能である. また, $G_{\mathrm{lb,FD}}(t)$ を時間 t について微分した

$g_{\mathrm{lb,FD}}(t)$ は次式で与えられる.

$$g_{\mathrm{lb,FD}}(t) = \begin{cases} 0 & \big(0 \le t < T_w\big), \\ C\exp\Big(-\alpha\big(t - T_w \cdot \log(t)\big)\Big) \times \alpha\Big(1 - \dfrac{T_w}{t}\Big) & \big(T_w \le t < D\big), \\ C_1 \cdot \alpha \cdot \exp(-\alpha t) & \big(D \le t\big). \end{cases} \tag{4.16}$$

$$\Big(C = \exp\big(\alpha \cdot T_w(1 - \log(T_w))\big), \quad C_1 = \exp(\alpha D) \cdot C \cdot \exp\Big(-\alpha\big(D - T_w \log D\big)\Big)\Big)$$

また, 4.3.1 の (4) と同様にして, 平均チェーン成長時間 T_{cg} の上限を次式で求めることができる.

$$\begin{aligned} \int_0^\infty t \cdot g_{\mathrm{lb,FD}}(t)\ dt &= \int_{T_w}^{D} t \cdot g_{\mathrm{lb,FD}}(t)\ dt + \int_D^\infty t \cdot g_{\mathrm{lb,FD}}(t)\ dt, \\ &= C \cdot \alpha^{-1-\alpha T_w}\Big(-\Gamma\big(\alpha T_w + 2, \alpha D\big) + \alpha T_w \Gamma\big(\alpha T_w + 1, \alpha D\big) \\ &\quad + \Gamma\big(\alpha T_w + 2, \alpha T_w\big) - \alpha T_w \Gamma\big(\alpha T_w + 1, \alpha T_w\big)\Big) + C_1 \cdot \Big(D + \frac{1}{\alpha}\Big) \cdot \exp(-\alpha D). \end{aligned} \tag{4.17}$$

$$\Big(C = \exp\big(\alpha \cdot T_w(1 - \log(T_w))\big), \quad C_1 = \exp(\alpha D) \cdot C \cdot \exp\Big(-\alpha\big(D - T_w \log D\big)\Big)\Big)$$

図 4.10 に, D を操作変数とした上で, 平均チェーン成長時間 T_{cg} の上限 (式 3.9, 式 4.11, 式 4.17) をプロットした.

4.4.2 T_{cg} の下限の導出

最初に, $F(t)$ の上限 $F_{\mathrm{ub,FD}}(t)$ を求めるため, ブロック伝搬遅延の上限 D であること, 関係式 を用いて, $F(t)$ を評価する. この際, $F(\omega_\delta) = \delta (0 \le \delta < 1)$ とする.

$$\begin{aligned} T_w &= \int_0^\infty (1 - F(t))dt, \\ &= \int_0^{\omega_\delta} (1 - F(t))dt + \int_{\omega_\delta}^{D} (1 - F(t))dt, \\ &\le \omega_\delta + (D - \omega_\delta) \cdot (1 - \delta). \end{aligned} \tag{4.18}$$

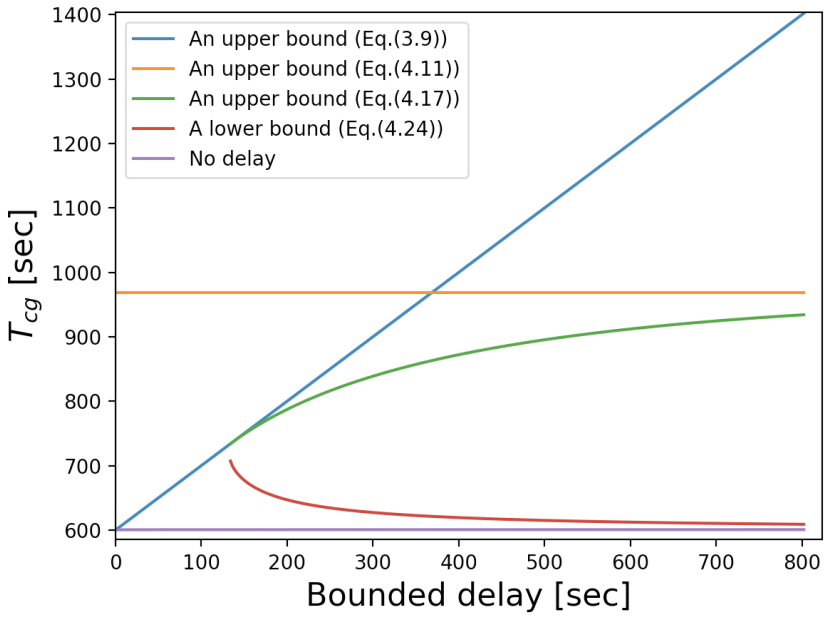


図 4.10: 平均チェーン成長時間 T_{cg} の上限, 下限をプロット. $\alpha = 1/600$, $P_F = 0.2$.

式 4.18 を変形すると, 次式が得られる.

$$\delta \leq 1 - \frac{T_w - \omega_\delta}{D - \omega_\delta}. \tag{4.19}$$

式 4.19 より, $F(t)$ の上限 $F_{\mathrm{lb,FD}}(t)$ が次式で計算できる. この時, 有界遅延モデルの条件である $F(t) = 1$ $(D \leq t)$ に注意する.

$$F_{\mathrm{ub,FD}}(t) = \begin{cases} 1 - \dfrac{T_w - t}{D - t} & \big(0 \leq t \leq T_w\big), \\ 1 & \big(T_w \leq t\big). \end{cases} \tag{4.20}$$

微分方程式 4.7 について, $F_{\mathrm{lb}}(t)$ の代わりに $F_{\mathrm{ub,FD}}(t)$ を代入して解く. この結果, フォーク確率と有界遅延モデルを考慮した上で, かつフォークが発生しないという条件に従う $G(t)$ の上限 $G_{\mathrm{ub,FD}}(t)$ が次式で与えられる.

$$G_{\mathrm{ub,FD}}(t) = \begin{cases} 1 - \left(\dfrac{D}{D-t}\right)^{\alpha(T_w - D)} & (0 \leq t < T_w), \\ 1 - \left(\dfrac{D}{D-T_w}\right)^{\alpha(T_w - D)} \cdot \exp(-\alpha(t - T_w)) & \big(T_w \leq t\big). \end{cases} \tag{4.21}$$

ここで, $G_{\mathrm{ub,FD}}(0) = 0$ 及び, $t = T_w$ における $G_{\mathrm{ub,FD}}(t)$ の右側極限と左側極限が一致することを初期条件とした. また, $G_{\mathrm{ub,FD}}(t)$ を時間 t について微分した $g_{\mathrm{ub,FD}}(t)$ は次式で与えられる.

$$g_{\mathrm{ub,FD}}(t) = \begin{cases} \dfrac{-\alpha(T_w - D)}{D} \cdot \left(\dfrac{D}{D-t}\right)^{\alpha(T_w - D)+1} & (0 \le t < T_w), \\ \left(\dfrac{D}{D-T_w}\right)^{\alpha(T_w - D)} \cdot \alpha \cdot \exp(-\alpha(t - T_w)) & (T_w \le t). \end{cases} \tag{4.22}$$

式 4.22 に t を掛け, $t = 0$ から ∞ まで積分した値は, フォークしないケースのみを考慮した平均チェーン成長時間 T_{cg} の下限 $T_{\mathrm{lb,nf}}$ となる.

$$\begin{aligned} T_{\mathrm{lb,nf}} &= \int_0^{\infty} t \cdot g_{\mathrm{ub,FD}}(t)\ dt = \int_0^{T_w} t \cdot g_{\mathrm{ub,FD}}(t)\ dt + \int_{T_w}^{\infty} t \cdot g_{\mathrm{ub,FD}}(t)\ dt, \\ &= \frac{-\alpha(T_w - D)}{D(A^2 - 3A + 2)} \cdot \left(D^2 - D \cdot \left(\frac{D}{D-T_w}\right)^{A-1} \cdot \big(-(A-1)T_w + D\big) \right) \\ &\qquad + \left(\frac{D}{D-T_w}\right)^{\alpha(T_w - D)} \cdot \frac{\alpha T_w + 1}{\alpha}. \\ &\qquad \Big(A = \alpha \cdot (T_w - D) + 1\Big) \end{aligned} \tag{4.23}$$

式 4.23 は, $D \to T_w$ の時に, $T_{\mathrm{lb,nf}} = 1/\alpha + D$ となり, 有界遅延モデルに一致するため, 妥当性がある (式 4.20 において, $D = T_w$ とすれば, $t = D$ に全ノードにブロックが初めて伝搬されるシナリオとなる).

以上より, フォーク発生時の平均チェーン成長時間 T_{cg} の下限を, 最長チェーン成長に全計算リソースが常に使われているシナリオを前提として, $T_{\mathrm{lb,f}} = 1/\alpha$ とすれば, T_{cg} の下限は次式で得られる.

$$T_{cg} \ge (1 - P_F) \cdot T_{\mathrm{lb,nf}} + P_F \cdot T_{\mathrm{lb,f}}. \tag{4.24}$$

式 4.24 をプロットした結果を, 図 4.10 に示す.

4.5 議論

4.5.1 導出した下限 $G_{\mathrm{lb}}(t)$ と有界遅延モデルの下限 $G_{\mathrm{nd}}(t)$ の比較

図 4.6, 4.7, 4.8, 4.9 では, 広範囲で $G_{\mathrm{nd}}(t) > G_{\mathrm{lb}}(t)$ となっている (表 4.3, 表 4.4 を参照). この原因は, 以下の 2 つであると思われる. 式 4.4.2 から計算される加重平均遅延が, $F(t) = F_{\mathrm{lb}}(t)$ の場合, 無限に発散する. 「ブロック伝搬遅延」が常に定数 D であるネットワークでは, 平均加重遅延は D に等しい. つまり, D を誤って高い値で評価する程, $G_{\mathrm{nd}}(t) < G_{\mathrm{lb}}(t)$ となる領域の増加が予測される.

図 4.10 から分かるように, D が一定値を超えると, フォーク確率 P_F を用いて評価した平均チェーン成長時間 T_{cg} の上限が, 有界遅延モデルで評価した上限を下回る. これは, 広範囲で $G_{\mathrm{nd}}(t) < G_{\mathrm{lb}}(t)$ であることを意味する. この原因は, 有界遅延モデルの評価手法では, ブロックがマイニングされてから D 時間分のブロック生成速度の影響が無視されるためと考えられる. このことは, 例えば, 表 4.4 で, $G_{\mathrm{nd}}(t_1) = p$ と $G_{\mathrm{lb}}(t_2) = p$ $(0 < p < 1)$ となる時間 t_1 と t_2 を比較することで理解される.

以上より, 導出した下限は, D が大きくなる P2P ネットワークでは, 従来手法より値の大きい下限を広範囲で提供可能と考えられる. また, 計測困難な D を用いず, 容易に計測可能なフォーク確率 P_F を用いて計算可能という利点がある. また, フォーク確率と有界遅延モデルを考慮した下限 $G_{\mathrm{lb,FD}}(t)$ は, 必要とするパラメータは多いが, 図 4.10 から分かるように, 最も値の大きい下限を提供可能である. これは, 本下限の導出モデルでは, ブロックを伝搬し始めてから全てのマイナーに伝搬するまでのブロック生成速度の影響を考慮したためである.

4.5.2 提案した評価手法を現実に応用する際の留意点と限界

実際のネットワークでは, マイニングに成功したノードの P2P ネットワーク内における地理的な位置や, 計算リソースの大きさで, フォーク確率 P_F は毎ブロック高ごとに異なる. このため, 分析に利用するフォーク確率は, 各マイナーノードの計算リソースやブ

表 4.5: ブロック生成速度が, チェーン成長時間とフォーク確率に与える影響

Block rate	An upper bound (Eq. 4.11)	Fork probability (Eq. 4.11)
1/600	631.15	0.01000
1/300	327.45	0.01990
1/100	122.04	0.05852
1/15	29.61	0.33103
1/5	16.62	0.70062
1	9.81	0.99759

ロック伝搬特性を考慮し, 各ブロック高のフォーク確率の上限値を推定し, その値を新たな P_F として同様の分析を行うことが望ましい. しかし, 各マイナー由来のフォーク確率を評価する必要があるため, 推定のためのサンプリング数が減少するという問題が発生する. 従って, 少ないサンプル数でフォーク確率を適切に推定する方法の確立が望まれる.

しかし, 提案した下限 $G_{\mathrm{lb}}(t)$ は, フォーク確率とブロック生成速度のみを用いて計算可能な点で有用である. また, 本分析を通じて, ブロックチェーンネットワークがほぼ同期しているかをピアソン距離を指標として判断可能となった (節 (3) を参照).

本提案手法を, より現実に近いモデルとして分析するためには, マイナーの参加, 離脱プロセスを含み, ブロック生成速度 α が時間と共に変化するモデル [51] 等を考慮する必要がある.

4.5.3 ブロック生成速度, フォーク確率とチェーン成長時間のトレードオフ

式 4.4.2 で計算される加重平均遅延 T_w を用いることで, 式 4.4.2 よりフォーク確率 P_F を計算可能である. また, 加重平均遅延 T_w, ブロック生成速度 α を用いることで, 式 4.11 よりチェーン成長時間の上限が計算できる. ここで, フォーク確率 $P_F = 0.01$, ブ

ロック生成速度 $\alpha = 1/600$ とした場合の加重平均遅延 $T_w \approx 2.619$ をベースとし, ブロック生成速度 α を増加させた際の各種パラメータの変化を表 4.5 に示す.

ブロック生成速度 α の増加により, チェーン成長時間の上限は低下するが, フォーク確率は増加する. ブロックチェーンにおいてフォークが存在する場合, 該当ブロック高から一つのブロックをコンセンサスアルゴリズムにより取捨選択する必要がある. フォークするブロック高が連続することは, ブロックチェーンのセキュリティが低下したと解釈でき, セキュリティと処理性能のトレードオフとなる.

2021 年現在の Bitcoin のフォーク確率は, 0.0004 程度 [52] となっており, セキュリティを重視したパラメータ設計となっていることが分かる.

4.6 結論

ブロックチェーンのチェーン成長時間の分布を, 容易かつ低コストな方法で理論的に計算可能なことが望まれる. 本章では, PoW ブロックチェーンに着目し, 容易に観測可能なフォーク確率を用いて, 最長チェーン成長時間の CDF の下限を閉形式で導出した. また, この下限を用いて, ネットワークがほぼ同期しているかどうかを判断するための指標となるピアソン距離を求めた. ネットワークシミュレーションを行い, 本下限と従来の有界遅延モデルに基づく下限を比較した. 従来下限は, D が十分に小さい場合, 有用な下限であるが, 多数のブロック受信データを計測する必要がある.

本章では, ブロックチェーンシステムが抱える, ブロック伝搬遅延, ブロック生成速度, ブロックデータサイズのトレードオフに着目した上で最長チェーン成長時間に関する分析を実行した. PoW を採用していないブロックチェーンでも, これらパラメータは考慮する必要がある. また, 現在でも PoW を採用したグローバルに稼働しているブロックチェーンが存在していることから, 本提案手法のような PoW ブロックチェーンの成長振舞い評価可能な分析手法が今後も求められる.

第5章　オフチェーン技術である ORU と待ち行列モデル

本章では, ブロックチェーンの外側を利用するオフチェーン技術 ORU に着目し, オフチェーン上の「トランザクション承認時間」を評価する待ち行列モデルを提案する. ORU では, 複数のオフチェーントランザクションをまとめ, その要約データ (ステートルート) をオンチェーンストレージに保存する. この要約プロセスの頻度により, オンチェーンのトランザクション負荷, オフチェーン利用者の待ち時間が変わる. 本モデルを用いる事で, 要約プロセスの頻度を決定するパラメータを上記トレードオフを考慮した上で, 適切に決定できる.

本章の内容は, 第一著者として発表した論文 [28] をベースとしている.

5.1　オフチェーン技術である ORU について

ORU (Optimistic RollUp) は, ブロックチェーンのスケーラビリティ問題を解決するために提案された (1.2.2 を参照). ORU は, ブロックチェーンの外側 (オフチェーン) にある任意データの完全性 (Data integrity) を保証可能であり, メジャーなオフチェーン技術として知られている. 本章では, ORU を用いた場合のオフチェーントランザクション (OTX: Offchain Transaction) の承認時間を評価する.

ORU では, オフチェーンストレージを更新する複数の OTX を, オフチェーン管理者 (オペレータとも呼ばれる) が一つのオンチェーン用トランザクションにまとめる (図 5.1 を参照). このトランザクションにより, 最新のステートルートがオンチェーンストレージに保存される. オフチェーン上の各データは, ステートルートを用いたマークル証明

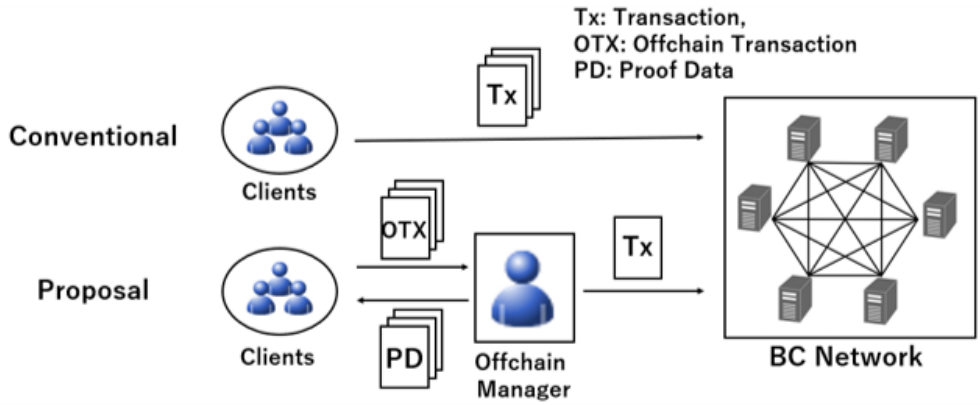


図 5.1: Rollup の概念図

[38] によって初めて完全性が保証される. 従って, オフチェーン利用者には, オフチェーン管理者から自身に関連したデータブロックに対応する証明データを受信するため, 待ちが発生する. この待ち時間を「オフチェーン利用者の待ち時間 $\mathrm{E}[W]$」と定義する. つまり, OTX 承認時間 $\mathrm{E}_{\mathrm{otx}}[T]$ は, オンチェーントランザクション承認時間 $\mathrm{E}_{\mathrm{tx}}[T]$ を用いて, $\mathrm{E}_{\mathrm{otx}}[T] = \mathrm{E}[W] + \mathrm{E}_{\mathrm{tx}}[T]$ となる ($\mathrm{E}_{\mathrm{tx}}[T]$ は, 節 3.2 で紹介した研究で評価済).

次節以降で, OTX をまとめる際のバッチサイズ b, オンチェーントランザクションのスループット TPS, 利用者の平均待ち時間 $E[W]$, 処理実行までのタイムアウト時間 T 等の各種トレードオフについて分析する.

5.1.1 ORU の類似したオフチェーン技術について

ORU と同様なオフチェーン技術に, ゼロ知識証明に基づく Rollup (zk-Rollup: zero-knowledge Rollup) がある. 両手法とも, 多数の OTX を一つにまとめ, バッチ処理結果のステートルートをオンチェーン上に保存する Rollup と呼ばれる手続きで, オフチェーンデータの完全性を保証する [53]. 正しく OTX を処理した結果得られるステートルートであることを保証するために, ORU は不正証明 (Fraud proof) [37] に基づく紛争解決プロトコルを利用し, zk-Rollup はゼロ知識証明を用いている.

しかし, zk-Rollup は, 暗号学的制約, Rollup 時の高い計算コストなどから, 実装が困

難な技術として知られている. 一方, ORU は, Rollup 時の暗号学的ハッシュ関数を用いたステートルート計算コストも低く, 複数システムの実装も行われている [15, 39].

5.2 Rollup 分析のための待ち行列モデルと数値解析

本節では, Rollup における, オフチェーン利用者の待ち時間 W, オンチェーントランザクションのスループット TPS, OTX のバッチサイズ b, 処理実行までのタイムアウト時間 T について分析する. 各パラメータの詳細は, 節 5.2.1 で説明する.

5.2.1 Rollup の待ち行列シナリオ

オフチェーン技術である Rollup のモデル化を実行するため, 以下に示す待ち行列シナリオを考える. 一人のオフチェーン管理者が, オフチェーンストレージを更新する複数の OTX を取りまとめて, ステートルートを時系列に並べる. その後, それらステートルートがリーフに割当てられるマークル木のルートハッシュと, 一番右のリーフに対応したステートルートをオンチェーンストレージに格納する. オフチェーンストレージ利用者は, このステートルートと自身のデータブロックに対応したマークル証明を, オフチェーン管理者から受信することで, オフチェーンデータの完全性を証明できる (節 2.3 を参照).

このステートルートと, マークル証明を受信するまでの平均待ち時間 $E[W]$ を計算するために, 待ち行列モデルを記述する. モデルを単純化するため, OTX の処理要求を受け, それらを要約したオンチェーントランザクションを送信するオフチェーン管理者が 1 人存在すると仮定する (図 5.1 を参照).

OTX は, レート λ のポアソンプロセスで到着して待ち行列に入り, その後, オンチェーントランザクション送信タイミングでバッチ処理される. バッチサイズは b に等しい. 待ち行列内の OTX 数が b に等しいか, タイムアウトが発生したとき, オンチェーン上の処理を実行するトランザクションが送信される (タイムアウト時間は $T > 0$ とする). 本論文では, このタイムアウト事象を「早期停止」と定義する.

オンチェーントランザクション送信後, オフチェーン管理者はサービスを停止し, 次のOTX が到着するまでサービスを停止する. この待ち行列モデルでは, オフチェーン管理者とオフチェーン利用者間の通信遅延を考慮しない. これは, 両者間で交換されるデータサイズが大きくないためである.

5.2.2 オフチェーン利用者の平均待ち時間評価

本節では, オフチェーン利用者がオフチェーン管理者から, マークル証明データを受け取るまでにかかる平均待ち時間 $E[W]$ を導出する. t を与えられた期間 $[0, T]$ におけるサービス開始からの経過時間とする. $t = 0$ はサービス開始時刻であり, $t = T$ は OTX が既定の数 b に達せず, タイムアウトが発生する時刻である. $N(t)$ を時刻 t において, オフチェーン管理者のキューに存在する OTX 数とする.

期間 $[t, t+dt)$ において, OTX 処理要求数が k である確率 $P_k(t)dt$ を次式で表す（$P_k(t)$ は確率密度関数 (PDF: Probability Density Function) である).

$$P_k(t)dt = \Pr\{N(t) = k\}. \tag{5.1}$$

5.2.1 で説明した待ち行列シナリオと, 上記の表記から, 以下の微分差分方程式 (Differential-difference equations) が得られる.

$$\begin{cases} \dfrac{\partial P_k(t)}{\partial t} = -\lambda P_k(t) + \lambda P_{k-1}(t), & 1 < k < b, \\ \dfrac{\partial P_k(t)}{\partial t} = -\lambda P_k(t), & k = 1. \end{cases} \tag{5.2}$$

右辺の第一項は, 微小時間 $[t, t + dt)$ において, キュー内の OTX 数が k から $k + 1$ まで増加する確率であり, 第二項はキュー内の OTX 数が $k - 1$ から k に増加する確率である.

さらに, 以下の境界条件が与えられる.

$$\begin{cases} P_k(0) = 0, & 1 < k < b, \\ P_k(0) > 0, & k = 1. \end{cases} \tag{5.3}$$

また, 次の正規化条件も与えられる.

$$\sum_{k=1}^{b-1} \int_0^T P_k(t)dt = 1. \tag{5.4}$$

微分差分方程式 5.2 を 式 5.3, 式 5.4 を用いて解く事で, 次の解が得られる.

$$P_k(t) = P_1(0) \cdot \frac{(\lambda t)^{k-1}}{(k-1)!} \cdot \exp(-\lambda t) \qquad (1 \leq k < b). \tag{5.5}$$

ここで, $P_1(0)$ は次式で与えられる定数である.

$$\begin{aligned} P_1(0) &= \left(\sum_{k=1}^{b-1} \int_0^T \frac{(\lambda t)^{k-1}}{(k-1)!} \cdot \exp(-\lambda t)dt \right)^{-1}, \\ &= \left(\sum_{k=1}^{b-1} \frac{1}{\lambda} \cdot \big(1 - \frac{\Gamma(k, \lambda T)}{\Gamma(k)}\big) \right)^{-1}. \end{aligned} \tag{5.6}$$

$\Gamma(a)$ と $\Gamma(a, x)$ はそれぞれガンマ関数と第二種不完全ガンマ関数 [54] で, 次のように定義される $(a > 0,\, x \geq 0)$.

$$\begin{cases} \Gamma(a) = \displaystyle\int_0^\infty t^{a-1} \exp(-t)dt, \\ \Gamma(a, x) = \displaystyle\int_x^\infty t^{a-1} \exp(-t)dt. \end{cases} \tag{5.7}$$

図 5.2 は, $\lambda = 1.0, T = 5.0,\, b = 4$ の場合の PDF の例である. 式 5.5 を用いて, 実行中のサーバーのキュー内の平均 OTX 数 $\mathrm{E}[N_{qr}]$ を次式により計算できる.

$$\begin{aligned} \mathrm{E}[N_{qr}] &= \sum_{k=1}^{b-1} \int_0^T k \cdot P_k(t)dt, \\ &= \sum_{k=1}^{b-1} \frac{kP_1(0)}{\lambda} \cdot \big(1 - \frac{\Gamma(k, \lambda T)}{\Gamma(k)}\big). \end{aligned} \tag{5.8}$$

一方, 停止中のサーバー のキュー内の平均 OTX 数は 0 である. 従って, サーバーの各平均実行時間 $\mathrm{E}[T_{qr}]$ とサーバーの平均停止時間 $\mathrm{E}[T_{qs}] = 1/\lambda$ を用いた加重平均から, サーバー内の平均 OTX 数 $\mathrm{E}[N_q]$ を次式で導出できる.

$$\begin{aligned} \mathrm{E}[N_q] &= \frac{\mathrm{E}[N_{qr}] \cdot \mathrm{E}[T_{qr}] + \mathrm{E}[N_{qs}] \cdot \mathrm{E}[T_{qs}]}{\mathrm{E}[T_{qr}] + \mathrm{E}[T_{qs}]}, \\ &= \frac{\mathrm{E}[N_{qr}] \cdot \mathrm{E}[T_{qr}]\lambda}{\mathrm{E}[T_{qr}]\lambda + 1}. \end{aligned} \tag{5.9}$$

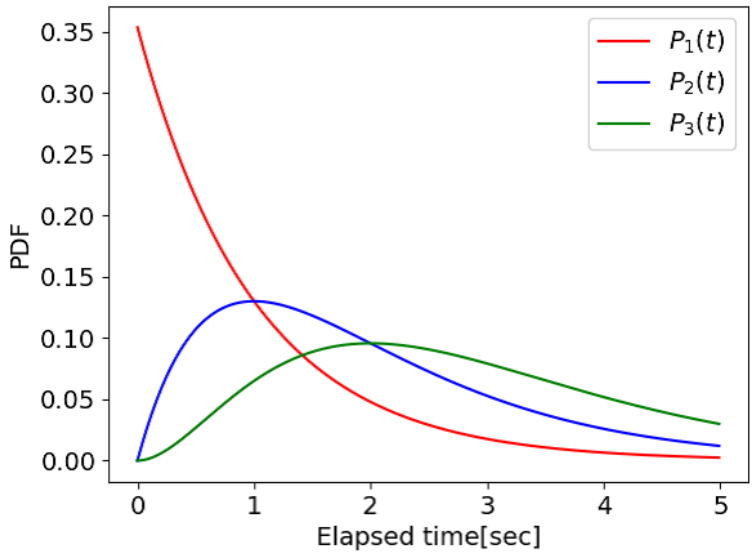


図 5.2: オフチェーン管理者のキュー内に存在する OTX 数の PDF の例 ($\lambda = 1.0, T = 5.0, b = 4$)

$\mathrm{E}[T_{qr}]$ は 5.2.3 の式 5.15 を用いることで計算できる. 式 5.9 と Little の公式 [55] から, サーバー内 OTX の平均滞在時間 (つまりは, 利用者の待ち時間 $\mathrm{E}[W]$) は, 以下のように計算される.

$$\mathrm{E}[W] = \frac{1}{\lambda} \cdot \mathrm{E}[N_q] = \frac{\mathrm{E}[N_{qr}] \cdot \mathrm{E}[T_{qr}]}{\mathrm{E}[T_{qr}]\lambda + 1}. \tag{5.10}$$

上式は, サーバーのタイムアウト時間 T と, バッチサイズ b 決定には適切でないことに注意する. 上式の導出では, 時間 T までに, OTX 数がバッチサイズ b に達するという事象に着目していないためである. 従って, 早期停止を考慮した $\mathrm{E}[W]$ の上限を以下のようにして導出し, T と b を決定するための指標とする.

(1) 早期停止を考慮した $\mathrm{E}[W]$ の上限

まず, オフチェーン利用者の平均待ち時間 $\mathrm{E}[W]$ に関する b に関する上限を導出する. このために, サーバーの待ち行列内の OTX 数がバッチサイズ b に達したときのみ, オン

チェーントランザクションを送信する, というシナリオを考える. 言い換えれば, $T \to \infty$ という条件の下で, サーバーが動作するシナリオである.

確率論に関する基本的な成果 [56] に基づくと, $i \geq 1$ 個の OTX がサーバーに到着する時間は, 形状パラメータ $\alpha = i$, 逆スケールパラメータ $\beta = \lambda$ のガンマ分布に従う. 従って, サーバー内の OTX 数が i となる, 時刻 t に関する PDF $g(t; i, \lambda)$ は, 次式で与えられる ($\int_0^\infty g(t; i, \lambda)dt = 1.0$).

$$g(t; i, \lambda) = \frac{\lambda^i}{(i-1)!} \cdot t^{i-1} \exp(-\lambda t). \tag{5.11}$$

タイムアウトが存在しないシナリオと仮定したため, サーバー内の OTX 数が k である確率は一様に分布する. つまり, サーバー内の OTX 数が k ($k = 0, 1, 2, ..., b-1$) である確率は $1/b$ である. このことから, オフチェーン利用者の平均待ち時間 $E[W]$ の上限 W_{ub} は, 以下の式で得られる.

$$\begin{aligned} \mathrm{E}[W] &\leq \sum_{k=1}^{b-1} \frac{1}{b} \cdot \int_0^\infty t \cdot g(t; b-k, \lambda)dt, \\ &= \sum_{k=1}^{b-1} \frac{1}{b} \cdot \frac{b-k}{\lambda}, \\ &= \frac{b-1}{2\lambda} \equiv W_{\mathrm{ub}}. \end{aligned} \tag{5.12}$$

上式の積分内部は, $k-1$ 個の OTX を保持するサーバーに到着した OTX (OTX_k と定義する) が, t 時間かけて $b-k$ 個の新たな OTX が到着した結果処理される確率密度を表している. つまり, これを 0 から ∞ まで t に関して積分すると, OTX_k の処理が完了する平均時間が得られる. 従って, $k = 1$ から $b-1$ について, OTX_k の生起確率 ($1/b$) と平均処理時間を考慮することで, 上限 W_{ub} を計算できる.

(2) シミュレーション値と理論値

オフチェーン利用者の待ち時間を評価するために, 5.2.1 で示した待ち行列モデルに基づいてシミュレーションを行った. 各シミュレーションは, 500 万件のオンチェーントラ

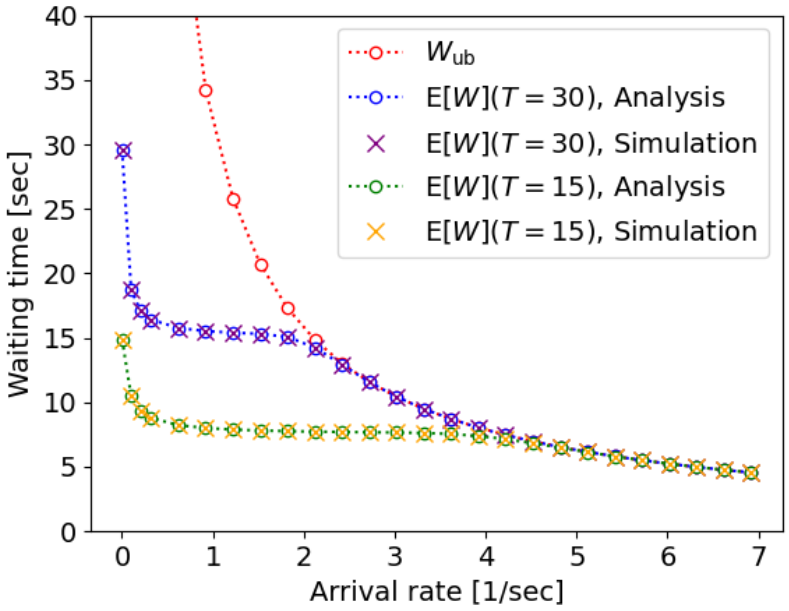


図 5.3: E[W] のシミュレーション値と理論値の比較 ($b = 64$)

ンザクションがオフチェーン管理者によって生成されるまで実行した. 図 5.3 は, バッチサイズ $b = 64$ の場合の例である. OTX の到着率 λ が低い場合, $[0, T)$ の期間に早期停止がほとんど起こらないので, W_{ub} は, タイトな上限となっていない. 一方, OTX の到着率 λ が高い場合, 早期停止が頻繁に起こるため, W_{ub} はタイトな上限となっている.

図 5.4, 5.5, 5.6 に, バッチサイズ $b = 32, 64, 128$ に関する各種理論値を示す. バッチサイズが大きいほど平均待ち時間が長くなるが, オンチェーン上の秒間当たりのトランザクション数は減少する (詳細は 5.2.4 で説明する).

5.2.3 バッチサイズ b とタイムアウト時間 T の決定方法

式 5.12 より, 設定したい待ち時間の上限 W_d と OTX 到着率 λ を用いることで, b の値を次式によって設定することができる.

$$b = 1 + W_d \cdot 2\lambda. \tag{5.13}$$

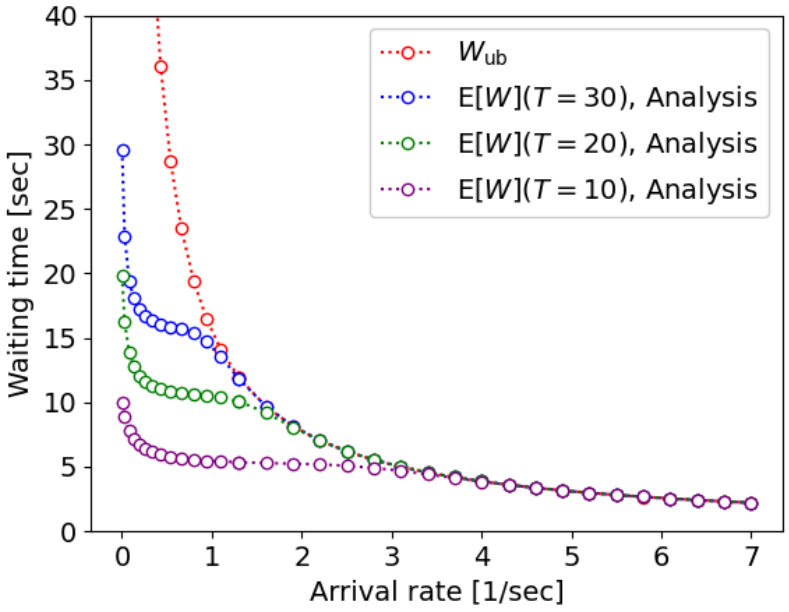


図 5.4: オフチェーン利用者の平均待ち時間の理論値 ($b = 32$)

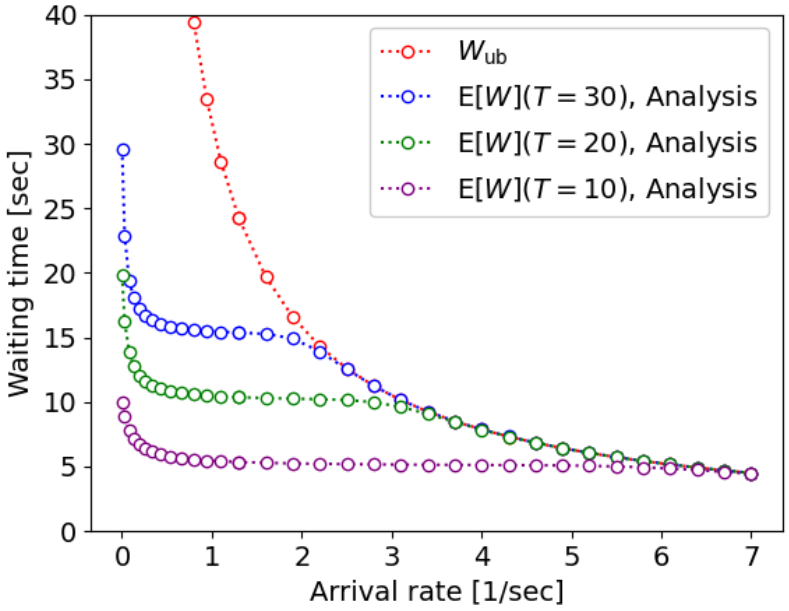


図 5.5: オフチェーン利用者の平均待ち時間の理論値 ($b = 64$)

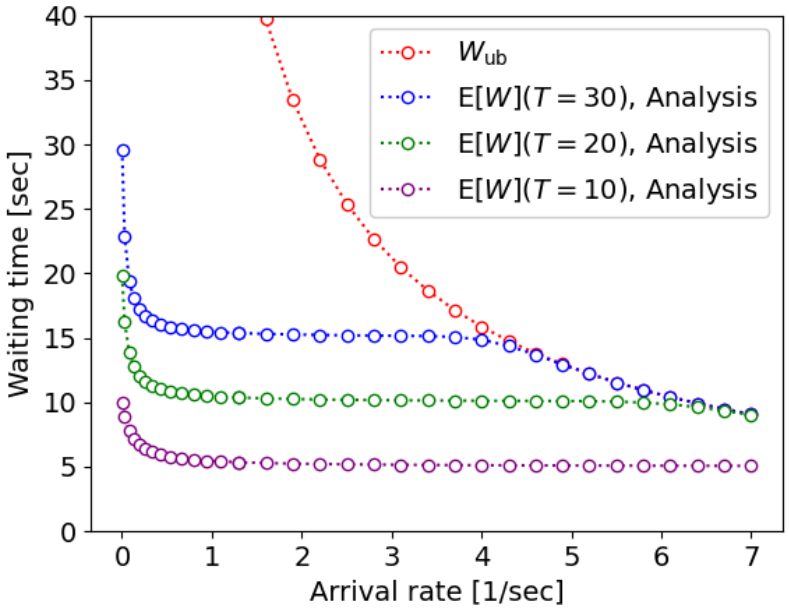


図 5.6: オフチェーン利用者の平均待ち時間の理論値 ($b = 128$)

また, タイムアウト時間 T とは早期停止が発生する確率 P_{ES} の観点からも決定することができる. 確率 P_{ES} は, 次式で求めることができる.

$$\begin{aligned} P_{\text{ES}} &= \int_0^T g(t; b-1, \lambda) dt, \\ &= 1 - \frac{\Gamma(b-1, \lambda T)}{\Gamma(b-1)}. \end{aligned} \tag{5.14}$$

従って, パラメータ P_{ES}, W_d, λ が与えられれば, 式 5.13 を用いてバッチサイズ b を決定し, 式 5.14 にバッチサイズ b を代入することで T を決定することができる.

また, 5.2.2 の式 5.9 で用いる, サーバーの平均実行時間 $\mathrm{E}[T_{qr}]$ は, P_{ES} を用いること

で次のように計算できる.

$$\begin{aligned}
\mathrm{E}[T_{qr}] &= \int_0^T t \cdot g(t; b-1, \lambda) dt + (1 - P_{\mathrm{ES}})T, \\
&= \int_0^T \frac{b-1}{\lambda} \cdot g(t; b, \lambda) dt + (1 - P_{\mathrm{ES}})T, \\
&= \frac{b-1}{\lambda} \cdot (1 - \frac{\Gamma(b, \lambda T)}{\Gamma(b)}) + (1 - P_{\mathrm{ES}})T. \qquad (5.15)
\end{aligned}$$

第一項は, 早期停止が発生した場合のサーバー実行時間に関する項であり, 第二項はタイムアウトが発生した場合のサーバー実行時間に関する項である.

5.2.4 オフチェーン利用者の待ち時間とオンチェーントランザクションレートのトレードオフ

本節では, バッチサイズ b と OTX の到着率 λ から, 秒間当たりのオンチェーントランザクション数 TPS を導出する.

5.2.1 で説明した待ち行列シナリオでは, サーバーは停止と稼働を繰り返す. このため, 各オンチェーントランザクションは, サーバーの平均稼働時間と平均停止時間毎に, ブロックチェーンネットワークに送信される. 従って, TPS は, 式 5.15 内のサーバーの平均稼働時間 $\mathrm{E}[T_{qr}]$ と平均停止時間 $1/\lambda$ 用いて次式で計算できる.

$$\begin{aligned}
TPS &= \frac{1}{\mathrm{E}[T_{qr}] + 1/\lambda}, \\
&= \left(\frac{b-1}{\lambda} \cdot (1 - \frac{\Gamma(b, \lambda T)}{\Gamma(b)}) + (1 - P_{\mathrm{ES}})T + 1/\lambda \right)^{-1}. \qquad (5.16)
\end{aligned}$$

また, オフチェーン管理者が常に最初の OTX が到着してから T 秒後にトランザクションを送信するシナリオを考えれば, TPS の下限を計算できる (このシナリオは, $b \to \infty$ を想定している).

$$\begin{aligned}
TPS &\geq \frac{1}{T + 1/\lambda}, \\
&= \frac{\lambda}{T\lambda + 1} \equiv TPS_{\mathrm{lb}}. \qquad (5.17)
\end{aligned}$$

TPS の理論値と下限を評価するために, シミュレーションを行った. 各シミュレーションは, オフチェーン管理者が 500 万件のオンチェーントランザクションを生成するまで実行した. 図 5.7 は, λ を変化させた場合に関する TPS のシミュレーション値を示している. λ が閾値を超えると, ($P_{\mathrm{ES}} \approx 1$ のとき) TPS が線形に増加する. 図 5.8 では, T が閾値を超えると ($P_{\mathrm{ES}} \approx 1$ のとき), TPS は定数に収束し, それ以上改善されないことがわかる. 従って, T を増加しても TPS が改善しない場合, バッチサイズ b を増加させる必要がある.

図 5.9 は, λ,b を変化させた場合に, TPS の理論値がどのように変化するかを示している. この図より, b を指数関数的に増加させることによって, TPS を指数関数的に減少させることが可能と分かる.

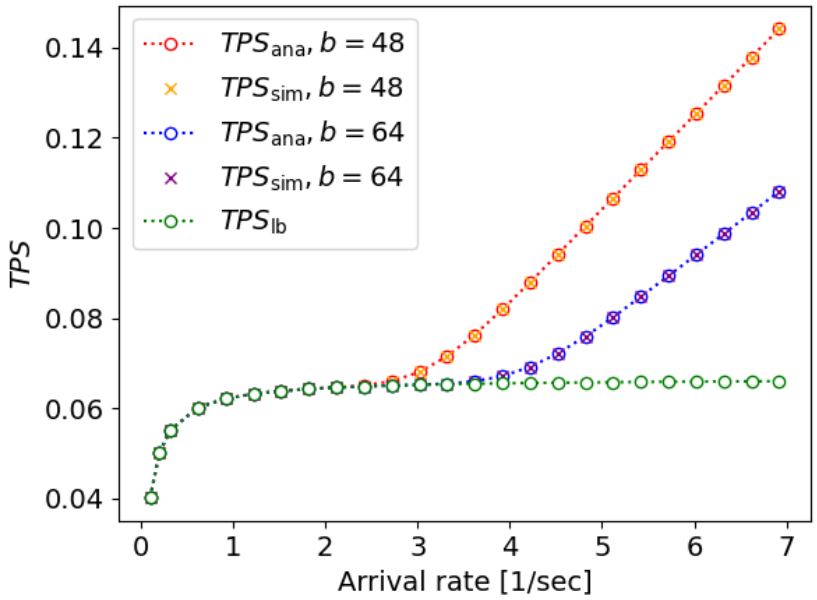


図 5.7: TPS の理論値とシミュレーション値の比較 ($T = 15$)

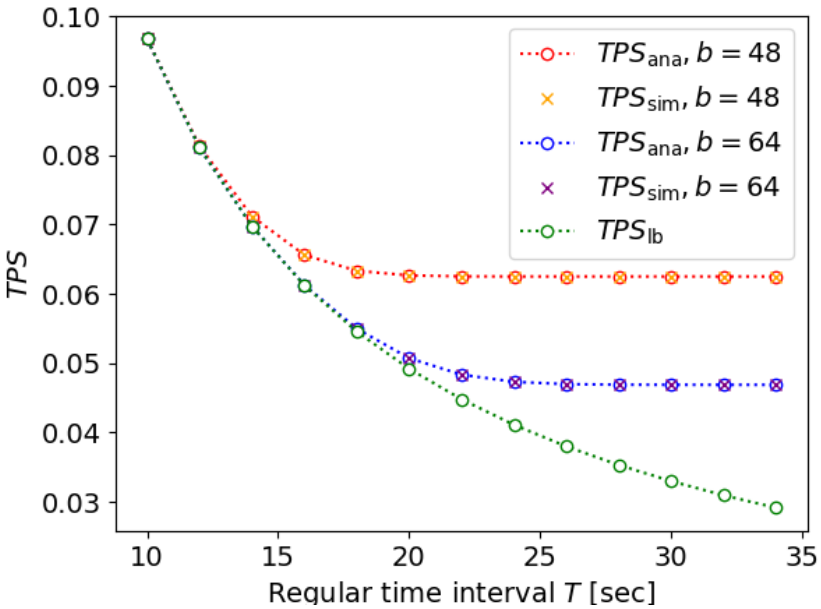


図 5.8: TPS の理論値とシミュレーション値の比較 ($\lambda = 3.0$)

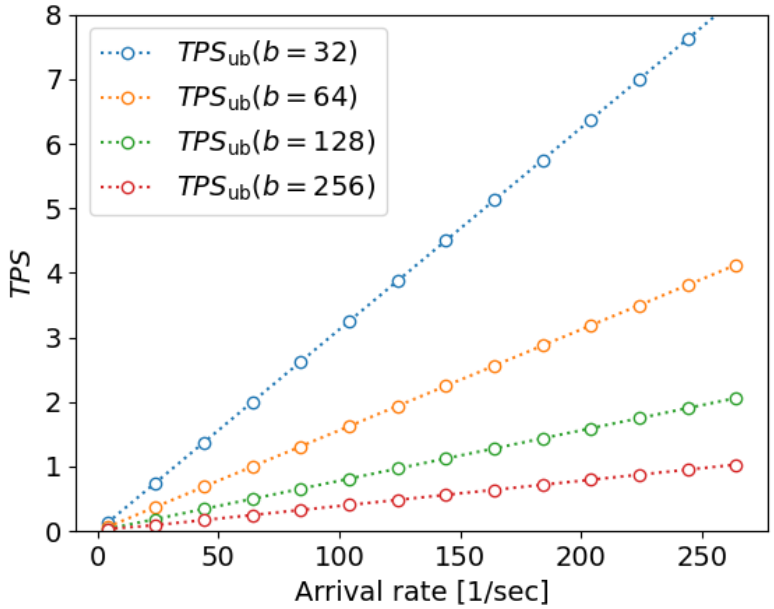


図 5.9: TPS の理論値と b を変化させた場合の影響 ($T = 30$)

5.3 提案した待ち行列モデルの分析

本節では, 提案した待ち行列モデルの持つ特徴, 及び実システム分析に応用する際の限界点について説明する.

5.3.1 オフチェーンユーザーの待ち時間の理論値と収束性

図 5.3, 5.4, 5.5, 5.6 が示すように, OTX 到着率 λ が増加するほど, 式 5.10 で与えられるオフチェーン利用者の待ち時間 $\mathrm{E}[W]$ が, 式 5.12 で与えられる上限 $W_{\mathrm{ub}} = (b-1)/2\lambda$ の値に収束する. このことは, 次のようにして確かめることができる.

式 5.15 で与えられる $\mathrm{E}[T_{qr}]$ は, $\lambda \to \infty$ において, 早期停止確率が $P_{\mathrm{ES}} \to 1$ となるため, $\mathrm{E}[T_{qr}] \approx (b-1)/\lambda$ のように近似できる. また, 式 5.8 で与えられる $\mathrm{E}[N_{qr}]$ は, $\lambda \to \infty$ において, $P_1(0) \approx (\sum_{k=1}^{b-1} 1/\lambda)^{-1} = \lambda/(b-1)$ と近似できるため, 次式のように近似できる.

$$
\begin{aligned}
\mathrm{E}[N_{qr}] &= \sum_{k=1}^{b-1} \frac{kP_1(0)}{\lambda} \cdot \big(1 - \frac{\Gamma(k, \lambda T)}{\Gamma(k)}\big), \\
&\approx \sum_{k=1}^{b-1} \frac{k}{\lambda} \cdot \frac{\lambda}{b-1} \cdot \big(1 - 0\big), \\
&= \frac{b(b-1)}{2\lambda} \cdot \frac{\lambda}{b-1} = \frac{b}{2}.
\end{aligned} \tag{5.18}
$$

従って, $\lambda \to \infty$ において, 式 5.10 で与えられるオフチェーン利用者の待ち時間 $\mathrm{E}[W]$ は次のように近似できる.

$$
\begin{aligned}
\mathrm{E}[W] &= \frac{\mathrm{E}[N_{qr}] \cdot \mathrm{E}[T_{qr}]}{\mathrm{E}[T_{qr}]\lambda + 1}, \\
&\approx \frac{(b/2) \cdot \big((b-1)/\lambda\big)}{\big((b-1)/\lambda\big) \cdot \lambda + 1}, \\
&= \frac{b-1}{2\lambda}.
\end{aligned} \tag{5.19}
$$

つまり, 早期停止が常に発生する ($P_{\mathrm{ES}} \approx 1$) 程に OTX 到着率 λ が増加した場合, タイムアウト時間 T がオフチェーン利用者の待ち時間に与える影響は無いといえる.

5.3.2　オンチェーントランザクションスループットの理論値と収束性

図 5.8 は, ある閾値を超えるとタイムアウト時間 T を増加させても, オンチェーントランザクションのスループット TPS が改善せず, ある一定値に収束することを示している. $T \to \infty$ とした場合に, タイムアウトが発生せずに常に早期停止する ($P_{\mathrm{ES}} = 1$) であることを考慮すると, 式 5.16 は次のように近似できる.

$$\begin{aligned} TPS &= \left(\frac{b-1}{\lambda} \cdot (1 - \frac{\Gamma(b, \lambda T)}{\Gamma(b)}) + (1 - P_{\mathrm{ES}})T + 1/\lambda \right)^{-1}, \\ &\approx \left(\frac{b-1}{\lambda} + 1/\lambda \right)^{-1} = \frac{\lambda}{b}. \end{aligned} \tag{5.20}$$

つまり, タイムアウト時間 T を変更した場合における, オンチェーントランザクションのスループットの改善の限界点 (つまりは, TPS の下限) を, 式 5.20 は表しているといえる.

5.3.3　提案した待ち行列モデルの妥当性

2022 年現在の Ethereum の Rollup では OTX の可用性を保証するために, 全 OTX をオンチェーン上に保存する仕組みを採用している. このため, オンチェーントランザクションは全 OTX 分のデータサイズだけ増加する. 従って, ブロック伝搬遅延の影響を考慮すると, オンチェーントランザクションのスループットのみでは, ブロックチェーンの「トランザクション承認遅延」の問題を緩和したかどうかを判断するためのパラメータとしては不十分の可能性がある.

また, 本提案モデルでは OTX の処理, 検証に関する遅延については, 十分短い時間で処理できるとして無視している. ORU と異なり, zk-Rollup では楕円曲線のペアリング演算といった高いコストの計算をオフチェーン管理者が実行する必要がある. このため, OTX の処理, 検証にかかる時間を考慮した待ち行列モデルの分析も今後必要である.

5.4 結論と今後の展望

Rollup では, オフチェーン管理者がオフチェーントランザクション (OTX) を一定数収集した後, オフチェーンストレージのデータ完全性を保証するステートルート情報を保持した一つのオンチェーントランザクションをブロックチェーンネットワークに送信する. 本章では, Rollup 分析用の待ち行列モデルを提案し, オンチェーントランザクションの送信時のタイムアウト時間 T, OTX のバッチサイズ b, オフチェーン利用者の待ち時間 E[W], オンチェーントランザクションのスループット TPS などを理論的に導出した. 各理論値の妥当性は数値シミュレーションによって再確認されている. 本提案モデルは, オンチェーンスループットとオフチェーン利用者の待ち時間とのトレードオフなどを考慮した上で, 各種パラメータの決定する際に有用である.

一方, 前節で述べたように, 本待ち行列シナリオでは, オンチェーントランザクション承認による遅延は考慮していない. ORU と異なり, zk-Rollup では OTX を実行する際に, オフチェーン管理者が高いコストの計算を実行する必要がある. 従って, 待ち行列シナリオにトランザクション処理実行時間を加えた分析が必要となる.

第6章　ケーススタディ 1: ucode 所有権管理システム

本章では, ブロックチェーンベースの ucode (ubiquitous code) 所有権管理システムをアプリケーションの一例として提案する. 現在の ucode 所有権は, DNS (Domain Name System) と同様に, 階層構造内の多数の信頼できる組織によって管理されている. 提案システムでは, ブロックチェーンの合意形成により ucode の所有権がユニークに定まる.

また, 本システムは, フロントランニング攻撃 [25, 26] を緩和するために, 二段階手続きによる ucode 所有権割当を行っている. この際, 節 4.1 で紹介したロック時間 (Lock time) 付きトランザクションを使用する. ロック時間付きのトランザクションのパラメータ設定は, 第 4 章で紹介した「チェーン成長の振舞い」の時間分布分析に基づき設定する.

本章の内容は, 第一著者として発表した論文 [29] をベースとしている.

6.1　背景

ユビキタスコンピューティングでは, 実世界の文脈を認識することが重要とされる. このため, 動的かつ柔軟な状況認識 (Context awareness) フレームワークを提供するために, ユビキタス ID (uID) アーキテクチャ [57] が提案されている. その基本的な考え方は, 実世界に存在する様々なモノ, 場所, 概念 (これらはエンティティと呼ばれる) を, 128 bit のユニークな識別子を用いて識別することである. 識別子は ucode (ubiquitous code) [57] と呼ばれ, ITU-T [58] と IETF [59] で標準化されている.

二つのエンティティに同一の ucode が割当てされないことを保証するために, 現在の ucode 所有権は DNS (Domain Name System) と同様に, 複数組織によって階層的に管

理されている．この従来型所有権管理システムでは, 下位組織に ucode 部分空間の管理を逐次的に委任する．つまり, 全ての ucode 所有権をセキュアにするためには, 階層内の全組織を信頼する必要がある．信頼できない組織が存在した場合, その組織の管轄する ucode 所有権情報は, 信頼できなくなる．

DNS において, 階層内組織を現実に起こった問題として, 200 万のドメインを管理していたレジストラ RegisterFly の倒産事件 [60] がある．RegisterFly が, 2007 年 3 月末に業務停止した際, 識別子であるドメイン所有権の管理が不十分であったために, ドメイン移行手続きに多大な労力を要したことが報告されている [60].

このような階層組織による所有権管理システムが抱える問題に対処するため, ドメイン名の所有権をブロックチェーン上で管理するシステム [61, 62, 63] が提案されている．本研究では, ucode の所有権を管理するブロックチェーンベースのシステムを提案する．

ブロックチェーン技術は, ディジタル通貨である Bitcoin [1] の土台技術として有名になり, 現在は金銭取引にとどまらず様々な分野での応用が期待されている．ブロックチェーンは, 信頼できる中央管理者を必要とせずに, P2P ネットワーク上で, 暗号学的に安全で一貫性のある, 分散データベース共有を可能にする．この性質により, 従来は信頼できる第三者を通じてのみ実現できたアプリケーションをブロックチェーン上で再構築する試みが多数行われている [64].

ドメイン名や ucode などの識別子の所有権割振りをブロックチェーン上で可能にするためには, [65] で言及されているスクワッティング (Squatting), フロントランニング (Front-running) 行為を抑止する仕組みが必要である．ブロックチェーンでは, トランザクション処理を中央集権的に制御できず, また「トランザクション承認遅延」が存在するため, この二つの問題が従来システムと比較して発生しやすい．

スクワッティング, フロントランニングを緩和するため, ブロックチェーンベースのドメインサービスを提供する Namecoin [61] と Blockstack [62] では, ドメイン登録時にコインを破壊する仕組みと, 2 段階のコミットプロセスによりドメイン登録を実行する仕組みを採用している．しかし, このドメイン登録手法では, 二回のトランザクションを決め

られた手続きでネットワークに送信する必要があり, ユーザーフレンドリーではないという指摘がされている [66].

Ethereum の分散型ネーミングシステムである Ethereum Name Service (ENS) [63] では, EIP 162 [67] で定義された Vickrey オークション [20] を用いたドメイン登録方式を採用している. このオークションベースのドメイン登録方式では, 最も高い金額を入札したユーザーが対象の名前を取得できるため, 原理的にスクワッティングやフロントランニングが発生しない. しかし, 動作の仕組みが複雑で, ドメイン登録時に最低でも 3 回のトランザクションが必要という問題がある.

本章では, ブロックチェーン上で ucode 所有権を管理するシステムを提案する. 本システムでは, 簡単な手続きと 1 回の取引で済むブロックチェーンベースの ucode 割振り方式を提案する. 本方式は, ucode 割振り時に衝突しないユニークな番号をオンチェーン上のデータから自動生成し, ucode フィールドの特定箇所に割り当てることで実現される. この方式は, 現在のブロックチェーンを用いたドメイン登録方式を単純に適用した ucode 割振り方式と比較して, ユーザーフレンドリーで効率的である.

従来の方法は, 複雑な手続きと最低でも 2 回のトランザクションが必要であった. また, Ethereum 上に構築された提案システムについて, ucode が割振りされるまでの期待実行時間と, 一連の処理を実行するためのユーザー待ち時間を, 待ち行列モデルによる数値シミュレーションを実行することで評価した. この結果から, ブロックチェーンネットワーク内のトランザクションが混雑している場合でも, 提案手法ではフロントランニングを実行されるリスクがなく, ユーザーの待ち時間もなく, ユーザビリティが高いことを確認した.

6.2 関連研究と従来手法の問題点

6.2.1 Namecoin

Namecoin [61] は, ブロックチェーンに基づくドメインネームサービス構築を最初に試みた取り組みである. Namecoin ブロックチェーン上のネイティブ通貨である 0.01 NMC を破壊 (Burn) することで, Namecoin ブロックチェーン上でドメイン名を登録することができる. コインの破壊は, 誰も秘密鍵を知らない暗号学的なアドレスに送ることで実現される. この仕組みを導入した結果, 登録したドメイン名の数に比例してスクワッター (Squatter) の所持金が減るので, スクワッティングの発生を緩和することができる. しかし, H. Kalodner ら [65] は, 0.01 NMC という金額の設定値が低すぎたため, Namecoin に登録されているドメイン名の大半がスクワッティングされていると述べている.

上記スクワッティング対策に加えて, Namecion では, 予約トランザクション (Pre-order transaction) と登録トランザクション (Registration transaction) の 2 種類のトランザクションを, 順番にネットワークに送信する 2 段階コミットプロセスをドメイン登録時に使用することにより, フロントランニングを緩和している. このドメイン登録手法は, 以下のように実行される.

1. 最初に, ドメイン登録ユーザーは, 名前のハッシュ値のみをブロックチェーンネットワークに公開する予約トランザクションを送信する.
2. 続いて, 予約トランザクションが, 連続した $m(m \geq 0)$ 以上のブロックによって承認された後, 同じユーザーが, その塩漬けハッシュ (Salted hash) の原像であるドメインを含む登録トランザクションを送信する.

この時の名前の状態遷移を図 6.1 の (a) に示す. Pre-order 方式では, フロントランナーは, 正規ユーザーと比較して少なくとも $m+1$ ブロック分だけ登録トランザクションを送信するのが遅れる. フロントランナーは, 自身の予約トランザクションをブロックに含め, 登録トランザクションを送信するまでに少なくとも m ブロック待つ必要がある

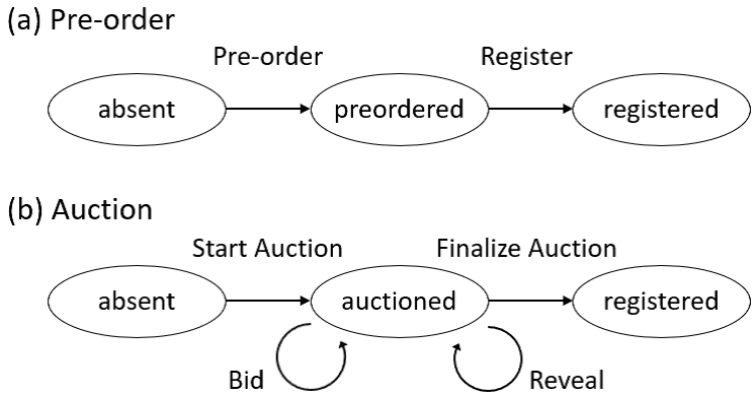


図 6.1: Pre-order 方式と Auction 方式を使用する場合のドメイン状態遷移図.

ためである. 従って, m が大きくなればなるほど, フロントランニング実行は困難になる.

6.2.2 Blockstack

Blockstack [62] は, Bitcoin によって保護されたグローバルなネーミングストレージシステムであり, スクワッティングやフロントランニングを防ぐために Namecoin とほぼ同じ仕組みを採用している[1]. Namecoin と Blockstack の大きな違いの一つは, Blockstack システムでは, ドメインを登録するために必要なコインの量が一定でないことである. 例えば, ドメイン名が短ければ短いほど, 登録者がそれを登録するために破壊する必要があるコインの量は多くなる. これは, 短い名前はより多くの人に望まれるという観察から着想を得ている [65]. その結果として, Blockstack はスクワッティングのリスクを Namecoin より適切に軽減している.

6.2.3 Ethereum Name Service (ENS)

ENS [63] は, イーサリアムブロックチェーン上に構築された分散型ネーミングシステムである. ENS は, EIP 162 [67] として定義されたブラインドオークションを利用した

[1] 2022 年現在, Blockstack は Stacks と呼ばれるようになっている.

オークションベースのドメイン登録方式を採用している．オークションは通常 5 日間開催され, 最初の 3 日間は入札に使用され, 次の 2 日間は入札金の公開に使用される[2]．入札時に, 入札金額はソルテッドハッシュ (Salted hash) [19] により隠蔽され, 入札金額以上の ETH のデポジットがロックされる．オークション期間が終了後に, 公開された中で最も高い金額を入札した落札者は確定トランザクションを送信し, 2 番目に高い入札金額をデポジットすることで名前を登録することができる[3]．

Auction 方式のドメインの状態変化を図 6.1 の (b) に示す．この方式では, 最高額入札者のみがドメインを取得できるため, 原理的にスクワッティングとフロントランニングの両方を防ぐことができる．

6.2.4 従来のドメイン登録手法の問題点

(1) Pre-order 方式

この方式を使う場合, 少なくとも連続した $m+1$ 個のブロックが生成されるための時間を待たなければならない．Namecoin の場合, $m = 11$ と設定されていて, Namecoin ブロックチェーンの平均チェーン成長時間が約 10 分であるため, 登録トランザクションを送信するまでの平均待ち時間は約 120 分となる．Blockstack のコミュニティでは, 予約トランザクションが m ブロックによって承認される前に, コンピュータをシャットダウンしてしまい, 登録トランザクションの送信に失敗したユーザーがいたことも指摘されている [66]．このような性質を持つため, Pre-order 方式はユーザーフレンドリーとは言えない．また, 図 6.2 のように, フロントランニング対策に失敗するケースが存在する．

[2] これらの期間は, Ethereum のタイムスタンプデータによって決定される．

[3] 最高額入札者が 1 人しかいない場合, 0.01 ETH をデポジットする必要がある． しかし, 2020 年に, Ethereum のトランザクション手数料が増加したことから, 0.01 ETH 引き出すのに 0.01 ETH 以上支払う必要がある状況になっていたことが指摘されている [68].

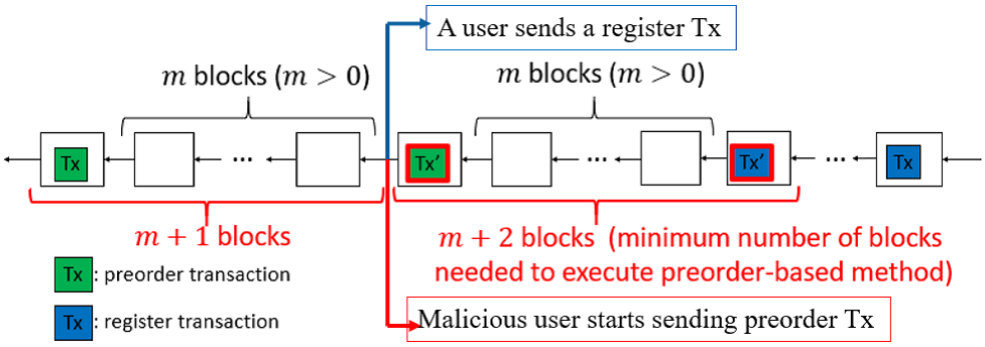


図 6.2: Pre-order 方式でフロントランニング攻撃対処に失敗する例

(2) Auction 方式

Auction 方式では, どのような名前であったとしてもオークション期間が終了するまで, ドメイン登録を実行することができない. 更に, この方式では入札者が 1 人であったとしても, 3 トランザクションを決められた手順に則ってネットワークに送信する必要がある. また, ドメインを取得したユーザーであるかどうかに関わらず, オークションに参加した入札者全員が入札に用いたデポジットを引き出す手続きも必要となっている.

6.3 提案した ucode 所有権管理システム

本節では, 提案する ucode 所有権管理システムの概要について説明する. また, 前節で説明した従来のドメイン所有権管理システムが抱えるドメイン登録手法の問題点を改善する ucode 割振り手法を提案する. 本提案システムは, Ethereum [3] 上のスマートコントラクトとして動作することを想定しており, ucode 割振りを行う時などに用いる実装時のパラメータについても説明する.

6.3.1 提案した ucode 構造と, 割振りされた ucode の所有権表現

現在の ucode 構造 [58, 69] は, 階層構造で ucode の所有権を管理するように設計されている. そこで, ブロックチェーンを用いた ucode 割振りに適した別の ucode 構造を導

表 6.1: 提案した ucode 構造

Field Name	**Length**
Version	4 bit
Ucode Allocation Method code (UAMc)	16 bit
Class Code (CC)	4 bit
Owner Level Domain code (OLDc)	n bit
Identification Code (IC)	$(104 - n)$ bit

入する. 提案する ucode 構造を表 6.1 に示す. 各フィールドの意味は, 以下で説明する.

Version (Ver):

本フィールドは, ucode のバージョン番号を表す. 現在の値は, ”0x0 ” である.

Ucode Allocation Method code (UAMc):

本提案手法で新たに定義したフィールドで, ucode 割振り方式を選択する際に使用する (選択可能な方式については後述する 6.3.3 および 6.3.4 で説明する). 現在の ucode 管理システムにおいては, このフィールドは TLDc (Top Level Domain code) として利用されている.

Class Code (CC):

n の値は CC の値 (例えば, $n = 8, 24, 40, 56, 72$ or 88) によって定まる. つまり, 本フィールドによって OLDc と IC の境界が決定される. 両フィールドの説明は以下の通りである.

Owner Level Domain Code (OLDc):

本システムでは, ucode は特定の管理組織を介さずに直接所有者に割り当てられる. 従って, 現在の組織識別コード (Second Level Domain Code: SLDc) は, OLD コードとして定義される ucode 所有者識別コードに置き換えることができる. すなわ

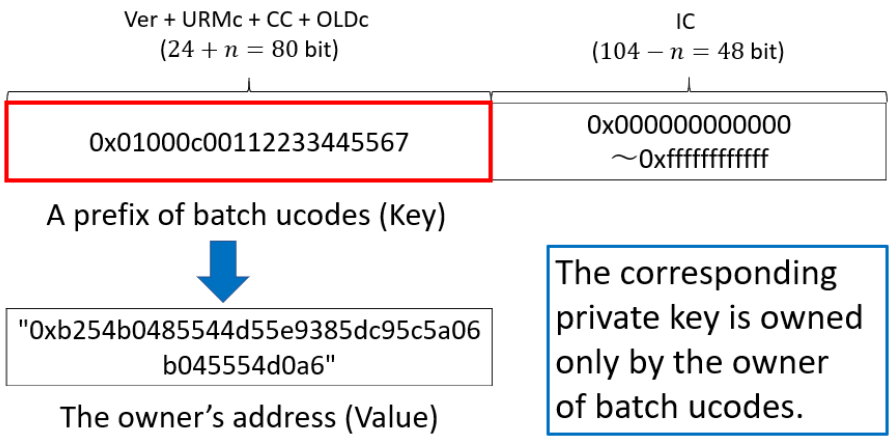


図 6.3: ucode の先頭 $24+n$ bit と所有者のアドレスの対応付け (例: $n=56$)．

ち, 全ての ucode に関して, その所有者は ucode の最初の $24+n$ bit によって一意に定まる (図 6.3 を参照).

IC (Identification Code):

本フィールドは, ucode の最初の $24+n$ bit で識別される ucode の所有者が, 残りの $104-n$ bit の ucode 空間を管理できることを示すものである．つまり, 提案した ucode 割振りが実行される度に, その所有者は 2^{104-n} 個のエンティティに ucode を割当てすることができることを意味する.

6.3.2 ucode 空間の大部分をスクワッティングする攻撃に対する対策

ucode は, 誰でも発行し利用することができるユニークな数値識別子である．その一方で, 多くの ucode を不正に占有する意図を持つ特定の利用者や組織への対策も重要である．このようなスクワッティング問題に対して, Namecoin, Blockstack, ENS などのブロックチェーンを用いたドメイン管理システムでは, ユーザーがドメインを登録する際に, ディジタル通貨の破棄やデポジットを要求する (節 6.2 の関連研究を参照)．つまり, スクワッティングに対するディスインセンティブを導入しているということである.

本提案システムでも ucode の割振り実行時にコインの破棄を必要とした. しかし, 破棄に必要なコインの量は非常に少なく, Blockstack と異なり一定に保たれるようにした. そのような設計をした理由は, 以下の 2 つである.

1. 全 ucode 空間には 2^{128}, つまり約 10^{38} の 3.4 倍の ucode が存在する. このように多数の ucode が存在するため, ucode のスクワッティングは原理的に困難である. そのため, ucode 割振り時に必要な価格を下げることができる.

2. 各 ucode はユニークな数値の識別子に過ぎず, 人間側の可読性 (Human-readable) が必要ない. したがって, 各 ucode の価格が等しいのは合理的である.

6.3.3 従来のドメイン割振り方式が応用される ucode 割振り方式

本システムでは, ucode の先頭 $24+n$ bit を所有者のアドレスに関連付けることで, 残りの $104-n$ bit の ucode 空間を所有者に割振りすることが可能となる. ここで, 重要なことは数字は文字に含まれているため, ucode の先頭 $24+n$ bit は名前として扱えることである. このことから, PUAM (Pre-order Ucode Allocation Method) として, Pre-order 方式のドメイン割振りを ucode 割振りに簡単に応用することが可能である. 同様に, オークションのドメイン割振り方式に関しても, Auction Ucode Allocation Method (AUAM) として簡単に応用可能である.

6.3.4 提案手法: Increment Ucode Allocation Method (IUAM)

ucode は, あくまでもユニークな数値識別子であり, 人間側の可読性は必要ない. つまり, 利用者の最も大きな関心は, バルクで割振りされた ucode 空間の大きさと一意性にある. そこで本提案では, ucode 割振り時に ucode の先頭 $24+n$ bit が一意に定まることのみを保証すればいい事に着目し, OLD code に自動的にシーケンス番号を割り当てる Increment Ucode Allocation Method (IUAM) を提案する. 本方式でインクリメント

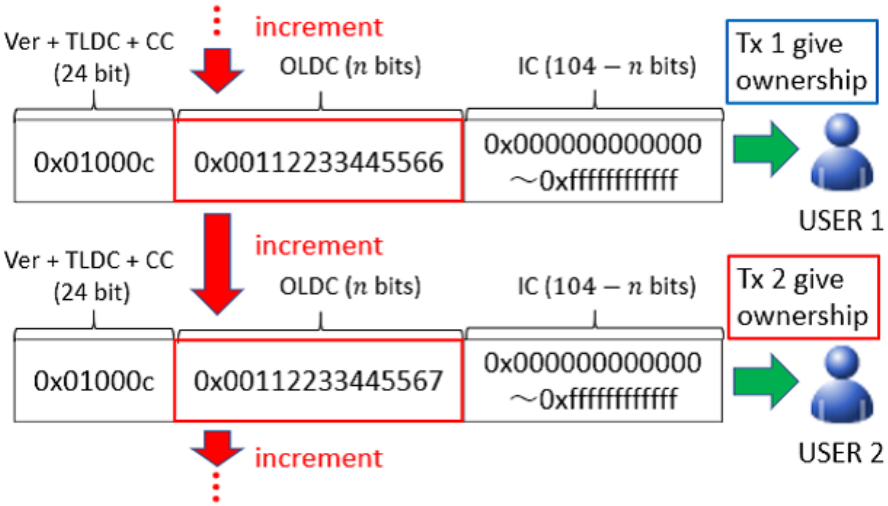


図 6.4: 提案した IUAM による ucode 割振り (e.g., $n = 56$).

キーとして扱われる OLD code は, MySQL における AUTO_INCREMENT 値のような役割を持つ [70]. 割振りされる OLD code の開始値は 0 であり, 図 6.4 のようにインクリメントトランザクションがブロックに含まれて, 処理される度に 1 ずつインクリメントされる. 割振りされる ucode の先頭 24 ビットは, 一意であるインクリメントキーによってユニークになるため, 割振りされる ucode 空間も一意となる. この処理における ucode の状態変化は図 6.5 のようになる.

IUAM は, 1 つのトランザクションと簡単な手続き （ユーザーが行うのはインクリメントトランザクションを送信するだけである) を行うだけで処理が完了するため, ユーザーフレンドリーで効率的である. また, IUAM はスクワッティングとフロントランニングの両方を防ぐことができる. スクワッティングの場合, 2^n の増分トランザクションを送信する必要があるが, 十分大きな n に対しては多くのコインを破壊する必要があるため, 現実的でない. また, 最新のインクリメントキーがオーバーフローしない限り, すべてのインクリメントトランザクションが常に一意の ucode を割振りするため, フロントランニングは理論的に不可能である. しかし, これらの良い点がある代わりに, 割振りする ucode の先頭 $24 + n$ bit を, IUAM ではユーザーが自由に選択することができない.

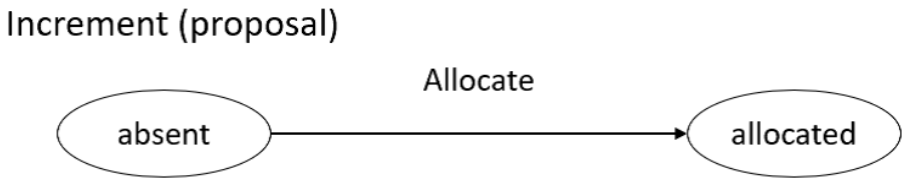


図 6.5: 提案した IUAM を用いた場合の割振りされる ucode の状態遷移

ユーザーが割振りされる ucode の先頭 $24+n$ bit を自由に選択したい場合, 6.3.3 で説明したような従来型の ucode 割振り方式を用いる必要がある.

6.3.5 ucode 所有権の譲渡方法

本提案システムでは, 利用者が ucode 所有権の譲渡を許可すれば, 他の利用者に ucode 所有権を譲渡することができる. 所有権の変更に受け取りての許可が必要な理由は, 悪意のある利用をされている ucode の所有権の一方的な譲渡を防ぐためである. そのため, 所有権の移転には, ucode の先頭 $24+n$ bit から生成されるハッシュと所有権を受け取る有効期限のタイムスタンプに対して, 受け取りての電子署名を必要とする. この署名は所有権を手放すユーザーが作成するトランザクションに含まれ, ブロックチェーン上で有効かどうかが検証される. この仕組みは, [65] 内のアトミックなドメイン転送と類似している.

6.3.6 Ethereum 上での実装のためのパラメータ

最初に, 本提案を実装する際のインフラストラクチャーとして, Ethereum を採用した 2 つの理由を以下で紹介する.

1. Ethereum はハッシュレートが高く, ディジタル通貨を取り扱える最も信頼性の高いブロックチェーンの 1 つである.

2. Ethereum のトランザクションは, Bitcoin のトランザクションが Bitcoin ブロック

チェーンに取り込まれるよりも早く Ethereum のブロックチェーンに取り込まれる．これは, 現在のイーサリアムの平均ブロック生成時間が約 14.7 秒だからである[4].

続いて, Ethereum [3] 上で提案システムのパラメータを設定する方法を解説する.

(1) OLDc サイズ n の設定方法

n の値は, 割振りされる OLDc に用いるインクリメントキーがオーバーフローする前に, 2^n 個のインクリメントトランザクションを実行できることを意味する．仮に $n = 24$ で, 各ブロックに 8 つのインクリメントトランザクションが含まれる場合, Ethereum の平均ブロック生成時間を考慮すると, 1 年でインクリメントキーが枯渇する事になる．この状況は現実に起こり得るため, $n = 40, 56, 72$ とすることが望まれる.

(2) 各 ucode の割振りに必要なコイン数の決定方法

1 回の ucode 割振りで破壊する必要のあるコインの量を, 本システムでは暫定的に 0.0001 ETH と設定した．2022 年 1 月現在, 0.0001 ETH の価格は約 0.38 ドルであり, スクワッターはブロックに含まれるトランザクション 1 回あたり少なくとも 0.38 ドルを失う必要がある (コインを破壊するため, トランザクション手数料を受け取ることができるブロック生成者も例外ではない)．ETH の価格は需要, 供給によって大きく変わる．また, 2021 年 8 月にトランザクション処理のユーザビリティを高めるために, Ethereum にブロック毎のベース手数料として ETH を破壊する EIP-1559 [71] がハードフォークにより導入された．このような社会的, 技術的な背景で ETH の価格は日々変化するため, 本提案システムでは, ENS のルートノード管理 [19] のように, 権限のある組織の複数署名によって ucode の割振り価格を変更できるような仕組みを導入した.

[4]Bitcoin の平均チェーン成長時間は約 10 分である.

6.4 Ethereum における各 ucode 割振り手法の平均承認時間評価のための数値シミュレーション

本節では, 各 ucode 割振り手法の平均承認時間を評価する. 従来手法を応用したオークション方式による ucode 割振り手法 (AUAM) は, ブロックチェーンのタイムスタンプを元に実行されるため, 平均承認時間はシステム設計に依存することになる. 従って, $E[T_{\mathrm{Pre-order}}]$ と $E[T_{\mathrm{Increment}}]$ として定義される PUAM と IUAM の平均承認時間を計算する. これを計算するために, トランザクション承認時間を T_{tx}, 連続した m 個のブロックを採掘する時間を S_m とする. また, $S_0 = 0$, $m \geq 0$ と仮定する. これら定義から, $E[T_{\mathrm{Preorder}}]$ と $E[T_{\mathrm{Increment}}]$ は, 次式で与えられる.

$$\begin{cases} E[T_{\mathrm{Preorder}}] & = E[T_{tx}] + E[S_m] + E[T_{tx}], \\ & = \mathrm{E}[S_m] + 2\mathrm{E}[T_{tx}]. \\ E[T_{\mathrm{Increment}}] & = E[T_{tx}]. \end{cases} \tag{6.1}$$

PUAM を完全に実行する時間は, 1 つの予約トランザクションが承認され, 連続した m ブロックが採掘され, 1 つの登録トランザクションを承認する時間と等しいので, 上記第 1 式が成立する[5]. また, IUAM の実行時間は 1 つのインクリメントトランザクションを承認する時間と等しいため, 上記の第 2 式が成立する. 以上から, $E[T_{tx}]$ と $E[S_m]$ が計算されれば, PUAM と IUAM の処理終了までの平均時間を評価可能である.

Bitcoin の平均トランザクション承認時間は, [13] 内の式 (17) で理論値が導かれているが, 最長チェーン成長時間 S_1 が独立同分布の指数変数であると仮定している. しかし, イーサリアムでは, ブロックの採掘難易度は各ブロックで調整され [3], ブロック伝搬遅延 [40] は平均ブロック生成時間に比べて無視できないほど大きいので, この仮定は成立しない. これら理由から, イーサリアムでは $E[T_{tx}]$ の理論値を得ることは困難である. そこで, $E[T_{tx}]$ を数値的に算出するために, [13] で記述されたマイニングプロセスを考慮し

[5]ブロックチェーンモデルを単純化するため, P2P ネットワーク上のブロックチェーン構築遅延とブロック伝搬遅延を無視した.

表 6.2: Ethereum において, m ブロックが生成されるまでの平均時間

Number of Blocks	**Average Time**	**Standard Deviation**
$m = 10$	146.66 sec	40.71 sec
$m = 20$	293.32 sec	58.68 sec
$m = 30$	439.97 sec	71.30 sec

た待ち行列モデルに Ethereum のブロック生成時間に関するブロック高 5,400,000 から 5,500,000 までの連続したタイムスタンプデータを適用することで, 数値シミュレーションを実施する. また, Ethereum の N 個のブロック生成時刻データを長さ m の連続した小周期 $k_1, k_2, ..., k_{\lfloor N/m \rfloor}$) に分割し, 周期時間の平均として $E[S_m]$ を算出すると表 6.2 に示すようになる.

6.4.1 $E[T_{Preorder}]$, $E[T_{Increment}]$ 計算のためのシミュレーション

$E[T_{tx}]$ を算出するために, 各ブロックの作成時刻としてイーサリアムのブロックタイムスタンプデータを使用する以外は [13] と同様のシミュレーションを行う.

(1) キューイングモデル

トランザクションはランダムなポアソン過程に従って到着する. 1 ブロックに含まれるトランザクションの最大数を b とする. システムに到着したトランザクションは, ブロックによってバッチ式に処理される. トランザクションがシステムに到着すると, そのトランザクションはまず待ち行列に入る. 現在採掘されているブロックが b より小さい取引を含んでいても, 到着した取引は常に後続のブロックに含まれ, 現在採掘されているブロックでは処理されない. このトランザクションの動作の詳細については [13] に記述されている.

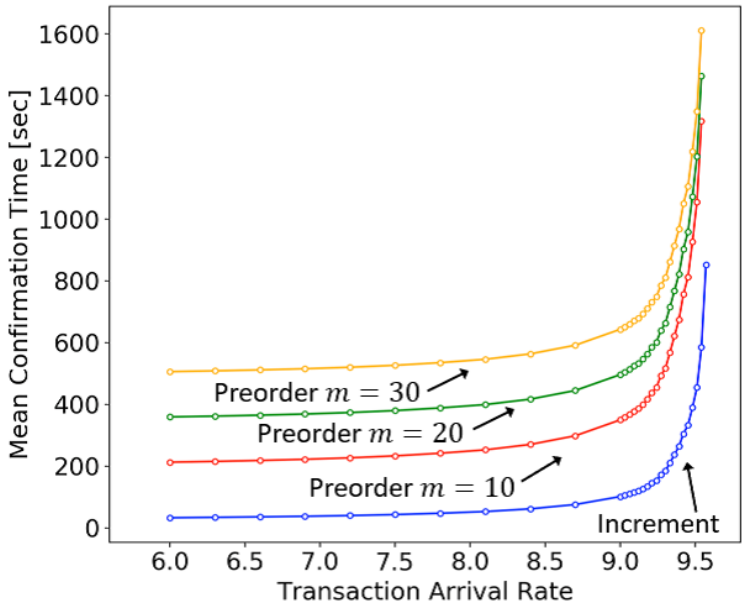


図 6.6: Increment vs. Preorder 方式による ucode 割振り比較 (Ethereum のブロック高 5,400,000-5,500,000 のデータを利用.

Ethereum では, b はガスと呼ばれる消費量によって制限される [3]. 本シミュレーションを行った 2018 年の各ブロックのガスの上限は 8,000,000 に設定されている. また, ブロック高 5,400,000 から 5,500,000 までの Ethereum のデータから, 1 取引あたりの平均 Gas コストは 56,042 である. これより, $b = 142$ と設定した. $\lambda E[S_1] < b$ を満たすとき, Ethereum ブロックチェーンシステムは安定である. よって, シミュレーション時の条件として, $\lambda < b/E[S_1] = 9.68$ を加える. これらの条件に従い, 待ち行列理論に従った離散イベントシミュレーションを実施した. このシミュレーションから得られた $E[T_{tx}]$ と表 6.2 内の $E[S_m]$ を式 6.1 に代入することで, 図 6.6 を得た.

6.5 議論

本節では, 各 ucode 割振り方法の比較を行う. 表 6.3 は 6.5.1~4 の内容を要約している.

表 6.3: 提案した ucode 割振り手法と従来手法の比較

	Number of Transactions	**Usability**	**Front-running Risks**	**Allocatable ucode Prefix**
Pre-order	2	×	Exist	Arbitrary
Auction	at least 3	△	None	Arbitrary
Increment	1	◯	None	Previous OLDc + 1

6.5.1 各 ucode 割振り手法の平均待ち時間

AUAM では, トランザクションの送信タイミングはオークション期間の設計に依存する. PUAM では, 登録トランザクションを送信する前に, ユーザーは少なくとも $m+1$ 個の連続したブロックが採掘されるまで待機しなければならない. このような特性から, Blockstack のコミュニティでは, ユーザーが待ち時間が終了する前に自身の PC をシャットダウンしたため, ドメイン登録に失敗したことを指摘している [66]. 一方, IUAM では, ユーザーは自身のインクリメントトランザクションが承認されるのを待つ必要がない. 言い換えるならば, IUAM では, ユーザーは自分のインクリメントトランザクションを送信した後に, すぐに PC をシャットダウンすることができる. これにより, IUAM のユーザビリティは PUAM より高いと言える.

6.5.2 PUAM を用いた場合のフロントランニングリスク

IUAM や AUAM を使用する場合, 理論上フロントランニングを行うことは不可能である. 一方, PUAM を用いると, 常にフロントランニングが発生しうる. フロントランナーが高い取引手数料を支払った場合[6], フロントランナーが待機しなければならない最小ブロック数は $m+2$ (予約トランザクションの承認に必要な 1 ブロック, PUAM で待

[6]デジタル通貨を扱うブロックチェーンでは, トランザクション手数料の高い取引は迅速に処理される可能性が高い.

機する必要がある m ブロック, 登録トランザクションの承認に必要な 1 ブロック）である. つまり, フロントランニングを防ぐためには, 少なくとも次の不等式が成立することが望ましい.

$$E[T_{tx}] < E[S_{m+2}]. \tag{6.2}$$

上式の左辺は, 節 6.4 のシミュレーションの場合, $\lambda \to b/E[S_1] = 9.68$ において無限に発散する. つまり, トランザクションが混雑しているとフロントランニングの実行が容易となる. これは, トランザクションのスループットが低いブロックチェーンにおいて, 十分に起こり得るシナリオであり, 極めて重要な問題である.

6.5.3 割振りされる ucode 空間の自由度

AUAM と PUAM は, OLDc がユーザー定義の ucode を割振りすることができる. しかし, IUAM では, ucode 割振り時の OLDc の値は, 1 つ前の ucode 割振りの OLDc の値に 1 を加算した値となるため, このようなことはできない. 従って, IUAM のみでなく, 必要な場合においては, AUAM, PUAM を併用することが望ましいと思われる.

6.5.4 Preorder 方式と, 時間制約付きトランザクション

Preorder 方式では, 予約トランザクションが承認されてから, m ブロック分経過後に, 登録トランザクションを送信する. 言い換えれば, 登録トランザクションは, 節 4.1 で説明したロック時間 (Lock time) 付きのトランザクションである. また, 登録トランザクションは, フロントランニングを完全に防ぐ (式 6.2 を満たすようにする) 場合, 有効期限 (Expiration time) 付きトランザクションであるとも言える.

ロック時間付きの登録トランザクションの待ち時間を $W_{tx} = S_m$ として, ブロック生成速度 $\alpha = 1/15$, フォーク確率 $P_F = 0.01$, $m = 20$ とすれば, 式 4.11 で計算される平均チェーン成長時間 T_{cg} の上限 T_{ub} を用いて, ロック解除の平均待ち時間の上限を計算

できる.

$$E[W_{tx}] = E[S_m] \leq m \cdot T_{ub} = 20 \cdot 15.78 = 315.6[\text{sec}]. \tag{6.3}$$

また, 有効期限付きと見做した登録トランザクションの平均承認時間 $E[T_{tx}]$ は, 平均チェーン成長時間 T_{cg} の下限 T_{lb} (式 4.24 等から導出) を用いて, 式 6.4 を満たせば十分である.

$$E[T_{tx}] < (m+2) \cdot T_{lb} = 22 \cdot 15.0 = 330.0[\text{sec}] < E[S_{m+2}]. \tag{6.4}$$

ここで, チェーン成長時間がチェーン生成速度と厳密に一致する同期モデルを考慮し, $T_{lb} = 1/\alpha = 15.0$ とした. このように, 第 4 章で行った「チェーン成長の振舞い」分析により, 時間制約付きトランザクションが満たすべき条件に付いて評価できる.

6.6 結論

階層構造を用いた複数の信頼できる組織が管理する現在の ucode 所有権管理は, コストが高い. そこで, 階層的な組織を必要としないブロックチェーン上で可能な ucode 所有権管理システムをデザインした. また, 提案システムでは, ucode に適した割振り手法である IUAM を提案した. 本方式は, 複雑な手続きと最低 2 回のトランザクションを必要とする現在のドメイン登録方式を単純に適用した ucode 割振り手法と比較して, 簡単な手続きと 1 回のトランザクションで済むため, ユーザーフレンドリーであり, 効率的である.

実証実験として, イーサリアムブロックチェーン上に構築した提案システムについてのケーススタディも行った. Ethereum で得られたブロックチェーンのデータに基づいた待ち行列シミュレーションを行い, 平均トランザクション承認時間を算出した. その結果、IUAM は割振りされる ucode の先頭 $24 + n$ bit をユーザー定義できないながらも、時間効率、ユーザビリティ、フロントランニングに対するセキュリティの面で優れていることが明らかとなった。

第7章　ケーススタディ 2: スケール可能な ucode 所有権割振り

本章では, 前章のブロックチェーンを用いた ucode (ubiquitous code) 所有権管理システムを, オフチェーンを用いることでスケールさせる手法を提案する. 提案した手法では, レジストラと呼ばれる主体がオフチェーントランザクションを 1 つにまとめることで, 複数のユーザーへの ucode 割振りを一括実行する. 各 ucode のリクエストデータはオフチェーン上に存在するが, オンチェーン上にあるデータと, ステートルートを用いたマークル証明により, 正統性が保証される. 各 ucode トランザクションをまとめる間隔やバッチサイズ等のパラメータは, 第 4 章で提案した ORU と待ち行列モデルを参考にして決定することができる.

本章の内容は, 第一著者として発表した論文 [28] をベースとしている.

7.1　背景

近年, IoT デバイスが世界中で広く利用されるようになり, グローバルな ID の需要が急速に高まっている. このため, スケーラブルで安全な識別システムを構築する必要がある. グローバル識別子の衝突を回避し, 識別子の所有権の正当性を証明するために, ブロックチェーン上での合意形成が有効である. しかし, 識別子の割振り要求数が増加した場合, ブロックチェーンのトランザクション処理性能 (スループットが低く, レイテンシーが高い) がボトルネックになる可能性がある.

本研究では, 混雑シナリオ下でも機能するスケーラブルなブロックチェーンベースの ucode 割振り手法を提案する. 本手法では, ucode の所有権取得を希望するレジストラ

ント (Registrant) は ucode リクエストデータをレジストラ (Registrar) に送信する．レジストラントは, 各 ucode リクエスト情報の整合性を保証するマークルツリーのルートハッシュを, 1 つのトランザクションを用いてオンチェーンストレージに保存する．そして, 対応する ucode 空間の所有権を証明するためのデータ (例えば, マークル証明など) を各レジストラに送信する．これより, レジストラントはオンチェーン上のセキュアにされたマークルルートと, 受信した証明データを用いて, 自身の ucode の所有権の正統性を証明することができる．また, オンチェーン上に保存されるマークルルートは固定長であるため, 割振り時のレジストラントの数に関わらず, オンチェーントランザクションのデータサイズは一定となり, ブロックチェーンの複製コストを削減することができる．この結果, 1 秒間に処理できる ucode リクエスト数は, 土台となるブロックチェーンの性能に制限されなくなる．

最後に, 証明データ受信のために発生するレジストラントの待ち時間と ucode 割振りスケーラビリティのトレードオフを分析した．ucode リクエストの到着率が指数関数的に増加しても, 1 つのトランザクションが処理できる ucode リクエストのバッチサイズを指数関数的に増加させることで, トランザクションスループットを十分に低減させることができる．

7.2 関連研究と目的

7.2.1 ブロックチェーンをスケールさせるオフチェーン技術

ブロックチェーンは検証可能な分散型データベースであり, 合意形成に参加する検証者は, 台帳の全てを複製し, 保持する必要がある．複製コストを削減によるブロックチェーンスケーリングのために, 土台となるブロックチェーンのプロトコルをベースにオフチェーンプロトコルを構築する．オフチェーンプロトコルでは, 土台となるブロックチェーン上の管理者による複製の対象外であるブロックチェーン外部のオフチェーンデータの整合性を保証する．この目的のため, Arbitrum [15] と TrueBit [39] は, ステートツリー (State

tree) に基づく証明メカニズムを使用している．ステートツリー [3, 37] はハッシュツリーの一種であり, 各リーフノードは, 対応するオフチェーンデータブロックに関する暗号学的ハッシュでラベル付けされている．ステートツリーの根 (Root) はステートルート (State root) [37] とも呼ばれ, ブロックチェーンの外側で動作するステートマシンの構成を一意に決定する．

ステートルートは時系列に並べられ, 暗号学的ハッシュ関数を用いることで, オンチェーン上においてもその順序が決定される．従って, 隣接する 2 つのステートルートとその最初のステートマシンの構成が与えられれば, 部分的なエミュレートをすることで誰でも状態遷移が正しいかどうか監査することができる．エミュレーションコストは, 計算機資源の限られたバリデータでも実行できるほど小さく設定される．その結果, すべての状態遷移を監視している一人の正直な検証者さえ居れば, 常に不正な状態遷移が行われた事を外部に対して証明することができる．

[72] の著者らは, Bitcoin のオンチェーンストレージのサイズを小さくするために, Trail アーキテクチャを提案した．Trail では, 各ノードはブロックのみを保持し, トランザクションや Bitcoin の UTXO [21], 口座残高を保存しない．アカウント残高はマークルツリー [38] によって保護され, そのルートハッシュはオンチェーントレージに保存される．Trail 上のライトクライアントがアカウント残高とその証明を保持することで, フルノードは各アカウントの残高を管理する必要がなくなる．このため, Trail は Bitcoin の 1/100 程度のブロック情報さえ保持していれば, 各トランザクションの検証を実行することができるようになる．

7.2.2 本研究の目的

前章で提案したブロックチェーンベースの ucode のような既存のブロックチェーンベースの識別システムは, ucode 割振りのスケーラビリティを考慮していない．本章では, ブロックチェーンベースのスケーラブルな ucode 割振り手法を提案する．また, 割振り要求数が増加した際に, 提案方式が機能するかどうか評価する．

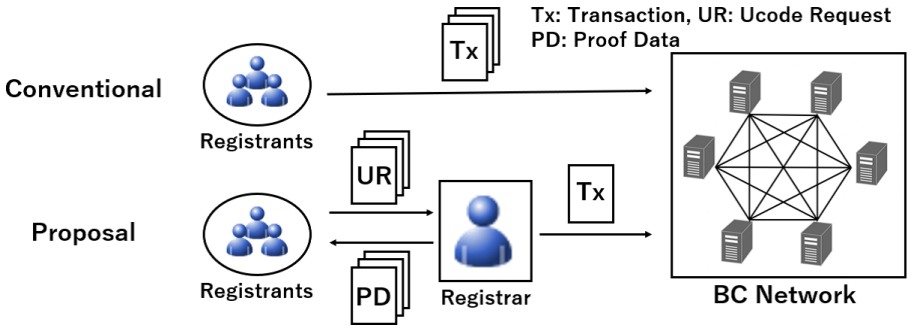


図 7.1: ブロックチェーンを用いた ucode 割振りの比較 (従来法と提案法)

図 7.1 に, ucode 割振りトランザクション数を減らすための基本的な考え方を示す. レジストラは, 複数の ucode の割振り要求をマークルツリーを用いて要約し, 1 つの固定データサイズのトランザクションとしてまとめる. マークル証明を用いる事で, ucode 割振り要求の内容の整合性は保証される. これにより, ブロックチェーン上の検証者のレプリケーションコストが削減される. しかし, レジストラントは ucode 所有権を主張するために, レジストラから証明データを受け取るのを待つ必要が出る. 第 5 章で提案した待ち行列モデルを用いて, レジストラントの待ち時間と ucode 割振りのスケーラビリティのトレードオフを分析する.

7.3 提案手法

本節では, 提案したブロックチェーンを用いたスケーラブルな ucode 割振り手法について説明する. 提案手法の構成要素を 7.3.1-4 で, ucode 割振りの全プロセスを 7.3.5 で説明する.

7.3.1 役割

提案した ucode 割振り手法では, 3 種類の役割が存在する.

1) 検証者**:**

検証者は, 基盤となるブロックチェーンを管理する存在である. 本システムでは, ブロックチェーンを用いることで, ブロックチェーン外側にある ucode の所有者情報の整合性を保護する. 詳細は 7.3.2 ∼ 5 で説明する.

2) レジストラ **:**

レジストラは, TLDc (Top Level Domain code) により識別されるエンティティである. レジストラントは, SLDc (Sub Level Domain code) で定義される ucode 部分空間を登録者に割り当てることができる. ucode の構造については, 7.3.2 で説明する. 本提案では, ブロックチェーンの複製コストを削減するため, レジストラはレジストラントからの ucode 割振り要求を 1 つのトランザクションにまとめる. また, レジストラは各レジストラントに対し, 対応する ucode 所有権に関する証明データを返信する. 図 7.1 に本処理の様子を示す.

3) レジストラント**:**

レジストラントは, ucode の所有権を得ようとするユーザーである. 本提案では, 参加者は ucode 割振り要求をレジストラに送信し, マークル証明データを受け取る. レジストラのトランザクションがブロックチェーン上で承認された後に, ローカルな証明データとオンチェーンストレージの関連データを用いることで, 自身の ucode 所有権が正当であると証明可能になる.

7.3.2 割振りされる ucode の所有権表現

表 7.1 に, 提案した ucode の構造を示す. ucode の先頭 20 bit が割振りを行ったレジストラのアドレス, 最初の $24 + n$ bit がレジストラントのアドレスと関連付けられる. つまり, ucode の TLDc は ucode を割振りするレジストラを特定し, ucode の SLDc はレジストラントを特定する. Class Code （CC) と Identification Code (IC) は, 通常の ucode と同じ役割をする. CC は, SLDc と IC の境界, つまり SLDc の bit 数 n を決定

表 7.1: 本提案における ucode 構造

Field Name	**Description**	**Length**
Version	A ucode version number.	4 bits
TLDc	Top Level Domain code.	16 bits
CC	Code that determines the boundary between SLDc and IC.	4 bits
SLDc	Sub Level Domain code.	n bits
IC	Code that indicates individual identification number.	$104 - n$ bits

する. IC は, 最初の $24 + n$ bit で識別される ucode の所有者が, 残りの $104 - n$ bit の ucode 空間を管理できることを意味する. 本論文では, CC までの上位 24 bit が等しい ucode 群をいかに効率よくレジストラントに割振りできるかに焦点を当てる.

7.3.3 ucode 所有権証明のための URD 木

本節では, ucode 所有権証明の際に用いる ucode 要求データ (URD: Ucode Request Data) 木を定義する. 図 7.2 に示すように, URD 木はマークル木の一つである. 各リーフノードには, 7.2 で定義される ucode 要求データブロックに一方向性関数を施したハッシュが与えられる. 各レジストラントは, URD, URD のハッシュ値, URD 木のルート, 及び自身の URD のマークル証明を提供することで, 自身の持つ URD の完全性を証明することができる. 図 7.2 は, 3 番目のリーフノードに属するレジストラントが, 青枠で示した 3 つのハッシュ値からなるマークル証明データを保持している例である.

7.3.4 ucode 所有者証明のためのオンチェーンデータ構造

提案手法では, ucode 割振りをスケールさせるために, 土台となるブロックチェーン上の検証者の複製コストを削減させる. 検証者は, 与えられた ucode 空間が複数のレジストラントによって所有されていることのみを検証する. 各 ucode 部分空間の所有権は, オ

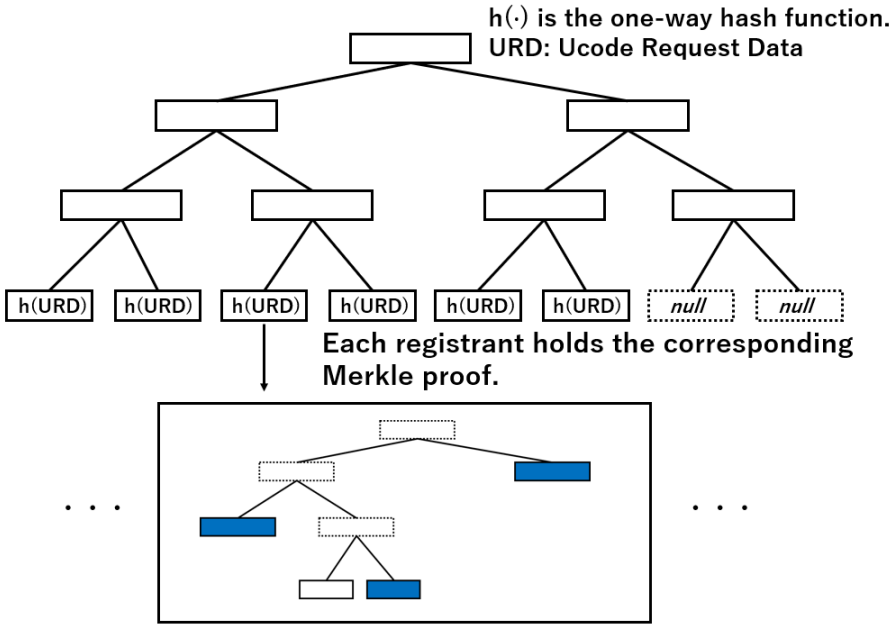


図 7.2: 登録者が対応する ucode 空間の所有権を主張できるようにするための URD ツリー

表 7.2: URD の構造. 最も左の葉ノードのインデックスは 1 と定義する. また, ディジタル署名の bit 長は, EIP 155 [73] を参考に決定した.

Field Name	**Description**	**Length**
Index	Index of the corresponding leaf node.	n bits
SLDc	Sub Level Domain code.	n bits
ucode prefix	The first 24 bits of allocated ucodes.	24 bits
Timestamp	Time when the registrant requests this ucode space.	256 bits
Owner's Address	The owner of this ucode space.	256 bits
Owner's Signature	A signature to prove the owner seeks this ucode space.	520 bits

表 7.3: オンチェーン上にある, ucode 所有権主張のためのデータ構造. x は, ucode 割振りトランザクションの総数である.

Field Name	**Description**	**Length**
ucode prefix	The first 24 bits of ucode space.	24 bits
Registrar's Address	The address of the registrar identified by the TLDc.	256 bits
List of Num of ucode Requests	The list about the number of ucode requests in the transaction.	n bits $\times x$
List of Highest SLDc Numbers	The list about the highest SLDc number in each transaction.	n bits $\times x$
List of Roots	The list of the URD tree roots.	256 bits $\times x$

ンチェーンストレージのデータとローカルなマークル証明を用いて, レジストラント各々が主張する. 表 7.3 に, ucode の所有権を証明するためのオンチェーンデータ構造を示す.

提案する ucode 割振りトランザクションが処理されるたびに, 表 7.3 中の 3 つのリストの末尾にデータが追加される. 最初のトランザクションでは, 最も高い SLDc 番号はそのトランザクションの ucode リクエスト数と等しい. 2 番目以降のトランザクションの最も高い SLDc 番号は, 前の番号に現在の ucode リクエスト数を加えたものになる. 本方式では, SLDc 番号が最新の最高 SLDc 番号以下の ucode 空間は, すでに割振りされたことになる（最初のトランザクションの場合, 前回の最高 SLDc 番号は 0 と見做す). URD 木のルートは, URD ブロックから一方向性ハッシュ関数により計算される.

7.3.5 提案したスケーラブルな ucode 割振り手法

本節では, 提案手法がどのように実行されるか解説する. 最初に, 各レジストラントは, インデックス, SLDc, 署名フィールドを除いた自身の URD データに関して署名を生成する (URD のデータ構造の参照 7.1). 次に, インデックスと SLDc フィールドを除いた自身の URD をレジストラに送信する. レジストラは, 図 7.2 のように, URD ブロックのハッシュ値を, レジストラから受け取った URD に対応するインデックスと SLDc を付与して並べることで URD 木を作成する. ここで, 各レジストラに割り当てられる SLDc 番号は, オンチェーンストレージ内の現在最も大きい SLDc 番号に対応するインデックス番号を加えたものになるようにする. これにより, 各インデックスの一意性がマークル証明により保証されるため, 割振りされた ucode SLDc の衝突も防ぐことが可能になる.

レジストラは各レジストラントに, 対応する URD ブロックのインデックスと SLDc フィールド, およびそのマークル証明を返送する. その後, レジストラはオンチェーンリストに 3 つのデータを追加するためだけの 1 つの割振りトランザクションを送信する. 表 7.3 より, 更新されるデータの総ビットは $2n + 256$ ビットに等しいので, この処理はブロックチェーンにおいても重くない. ブロックチェーン上でトランザクションが処理された後, 新しい最高 SLDc 番号が, 現在の番号に ucode リクエスト数を加えたものとして, オンチェーンリストに追加される. また, 現在の URD 木のルート, 現在の ucode リクエスト数 (これはマークル証明に用いられる) もオンチェーンストレージに格納される.

各レジストラは, URD とその Merkle 証明, URD ツリーと最高 SLDc 番号の対応するデータを用いて, 自身の ucode 所有権を主張する. URD の完全性とインデックスの一意性は, マークル証明により保証される. 従って, URD の SLDc 番号が対応する最高 SLDc 番号に URD のインデックス番号を加えたものと等しいことを確認すれば, 誰でも自分の SLDc がユニークなものであると確認することができる.

7.4 議論

7.4.1 分散化特性

本提案では, セキュアなオンチェーンデータにアクセスできる人なら誰でも, レジストラントから提示された ucode 所有権の証明が正当なものかどうかを判断できる. レジストラから一度与えられた ucode 所有権は, レジストラによっても決して無効化されず, 管理からも完全に独立している. 一時的に信頼できるレジストラが存在さえすれば, 提案手法は完全に機能する.

また, 本手法は, 少なくとも 1 つの信頼できる登録機関が存在する場合に, スケーラブルな ucode 割振り実行が可能である. 本手法において信頼できるレジストラを確立するためには, Casper [34] のようなセキュリティデポジットを用いることは特に有用である. なぜならば, 仮にレジストラが保証金を引き出しても, 既に割振りされた ucode の所有権が正当なものであることが保証されているからである.

提案する ucode 割振りは, 基盤となるブロックチェーン上にある検証者がレジストラントの署名を確認する必要がない. ucode の所有権が正当であることを確認する際, ディジタル署名の検証はローカルで行われ, オンチェーン上では行われないからである. ディジタル署名の検証は低負荷な操作ではない (例えば, Ethereum では 1 トランザクションあたり 21,000 Gas が必要に対して, ディジタル署名の検証では 3,000 Gas が必要である [3]）. この点も, オフチェーンベースの本方式の利点である.

7.4.2 提案手法と従来システムとの比較

Arbitrum [15] や TrueBit [39] などのステートツリーベースのオフチェーン技術は, ブロックチェーンの外部にある任意のデータを安全に保護することを可能にする. しかし, このようなシステムでは, オフチェーンストレージのすべての状態遷移を検証する正直な検証者が, 少なくとも 1 人必要となる. 正直な検証者がいなければ, ステートルートは信

頼できないものになり得る. つまり, 懐疑論者 (Skeptical) は Arbitrum や TrueBit を用いたブロックチェーン外部の状態遷移を一切信用できないのである.

一方, レジストラントのローカルストレージにある提案する ucode 所有権証明は, 常に誰からも信頼されるものである. これは, 各 URD の SLDc が, 最高 SLDc 番号と URD ブロックのインデックスによって一意に決定されるからである. 最高 SLDc 番号はコンセンサスアルゴリズムにより, URD ブロックのインデックスはマークル証明により暗号学的に安全性と一意性が保証されている. 従って, 本手法により, レジストラントはオンチェーンストレージのデータと同程度に安全な ucode 所有権データを, ブロックチェーンの外側に保持することが可能となる.

本システムでは, Trail [72] とは異なり, レジストラントにマークル証明の更新を要求しない. その理由は, 提案する URD 木は更新されないからである.

7.4.3 本手法の限界点

本提案では, 割振り可能な ucode の接頭辞 (Prefix) を自由に決定することについては考慮していない. 例えば, 本手法では, 特定の SLDc を持つ ucode を割振りする機能は持っていない. これは, [29] のオンチェーンベースの効率的な ucode 割振り手法でも実現されていない.

本手法はスケーラブルな ucode 割振りを可能にするが, 所有権移転や ucode 部分空間の再委任は考慮されていない. このような機能を持たせる場合, ucode 空間の所有権情報を安全に追跡する仕組みが必要となる.

7.4.4 第 5 章を用いた数値分析について

提案手法では, レジストラとレジストラント間で送受信するデータサイズは十分に小さい (表 7.2 を参照) ため, 「キューイング処理遅延」と比較して「通信遅延」は無視可能である. また, URD 木を構築する際に発生する遅延も十分に小さい.

従って, 第 5 章で提案した待ち行列シナリオ 5.2.1 に基づいて, レジストラントが証明データを受信するまでの待ち時間 $E[W_{ur}]$ を評価できる. 例えば, ucode リクエスト到着率 $\lambda = 0.05$, バッチサイズ $b = 32$, タイムアウト時間 $T = 600$[sec] とすれば, 式 5.10 を用いて, 以下のように計算できる.

$$\mathrm{E}[W_{ur}] = 286.73[\mathrm{sec}]. \tag{7.1}$$

7.5 結論

ユビキタスコンピューティングにおいて, グローバル識別子は重要な役割を担っている. グローバル識別子の衝突を防ぐためには, 信頼性の高い ucode 所有者情報を保持, 共有することが必要である. このため, 第 6 章で ucode 所有者管理システム [29] を提案した. しかし, 単位時間当たりの ucode 割振り数の上限という制約があった. この問題を解決するために, スケール可能な ucode 割振り手法を提案した. 本提案は, レジストラはレジストラントからの ucode 要求を 1 つのトランザクションに束ね, そのバッチサイズ b を大きくすることで, スケール可能な割当方法を実現した.

第8章　ケーススタディ 3: DNN モデルの生成プロセス検証

本章では, 2 章で解説したオフチェーンプロトコル ORU (Optimistic RollUp) を, DNN (Deep Neural Network) モデルの生成プロセスの検証に応用した際に, トランザクションの承認プロセスに関する影響ボトルネックについて評価を行った. ORU で用いられる紛争解決プロトコルには, 時間依存のロジックが含まれている. ブロックチェーンでは, ブロック提案者によるブロックタイムスタンプ操作が脆弱性として知られている [4]. このため, 一般的な紛争解決プロトコルでは, マイナーによる操作が困難であるブロック番号を利用して時間依存のロジックを処理している. ロジックで指定されたブロック番号に到達するまでの時間は, 第 4 章で解説したチェーン成長の振舞い分析によって評価可能である.

本章の内容は, 第一著者として発表した論文 [30] をベースとしている.

8.1　背景

近年, DNN (Deep Neural Network) は, 顔認識 [74], 画像分類 [75], 音声認識 [76], マルウェア検知 [77] などの幅広い分野で研究されている. その一方, 学習アルゴリズムの透明性欠如や, 学習データセットの起源が明らかでないといったセキュリティリスクのために, 実世界の重要なアプリケーションに応用することが困難な問題が指摘されている. 例えば, 学習中に一部のネットワークパラメータを悪意を持って操作するバックドア攻撃 [78, 79] は, DNN のセキュリティ問題としてよく知られている. AI Now 2019 Report [80] では, 機械学習研究者は AI の潜在的セキュリティリスクを考慮し, モデルやデータ

の出所を明文化しておくことが推奨されている. 2019 年のフォレンジックアルゴリズムの正義に関する法律 [81] において, 企業はソフトウェアの再現性を保証することが求められている. このような観点から, DNN モデルが生成されるプロセスを検証可能にするプラットフォームが必要と思われる.

しかし, 深層学習を実行するには膨大な計算リソースが必要である. このため, 計算リソースに制約のあるユーザー (以降ライトクライアントと呼ぶ) による学習プロセス全体の再検証は困難である. ライトクライアントは, 外部機関による学習モデルの検証結果を信頼せざるを得ないのである. 外部機関の判断の信頼性を担保するためには, 合意形成システムや紛争解決システムが必要となる.

ブロックチェーン技術は, S. Nakamoto の提案したディジタル通貨 Bitcoin の土台となっている. Bitcoin では, ディジタル通貨とその所有権に関する全ての状態遷移について合意形成を行うことで, ディジタル通貨の二重支払い問題を解決している [11]. Bitcoin システムの一般化を試みた Ethereum [3] では, ユーザー定義可能な任意の状態遷移について合意形成を行っている. 本章で応用先としている DNN の学習プロセスも, モデルのパラメータが決定論的に更新される場合, 状態遷移として記述することができる. つまり, DNN の学習プロセスはブロックチェーン上で原理的に検証できる.

しかし, ブロックチェーンの合意形成時間の間隔や, スケーラビリティの問題から現実的に困難である [11, 27]. このため, ブロックチェーンの外側 (オフチェーン) で実行された状態遷移計算の正しさを, ブロックチェーン上で実行される紛争解決プロトコルを用いて検証する手法 [15, 39] が提案されている. 紛争解決プロトコルにより, 一人でも正直な検証者が存在すれば, 誤った状態遷移の行われている部分をブロックチェーン上で指摘可能となる. この状態遷移計算コストは十分に小さくなるよう設定されているため, ライトクライアントでも紛争解決プロトコルにおける正しい主張を判定可能である.

本提案では, 深層学習プロセス全体をレイヤーベースの計算に分割し, 紛争解決プロトコルを実行可能にした. 各計算の検証に必要なデータ, 例えばニューロンの出力, 層間の重みとその勾配といったニューラルネットワークを記述する状態は, 暗号学的ハッシュ関

数によって一意に決定される．また, それらハッシュ群は複数のマークル木 [38] によって結合され, ルートハッシュがブロックチェーン上に保存される．不正な状態遷移により計算されるルートハッシュは, 正しい状態遷移を行ったことにより得られるルートハッシュと異なる．このため, 不正な状態遷移は, 正直な検証者により直ちに検知することが可能である．また, 二分法プロトコルにより, 不正な状態遷移が行われた箇所を提示可能にした．ライトクライアントは不正な状態遷移が行われた部分の計算を実行することで, 不正な状態遷移が行われたかどうかを確認することができる．

本研究では, 深層学習フレームワーク PyTorch を用いて, 提案した検証システムのプロトタイプを実装し, 紛争解決プロトコルを可能にする深層学習の実行時間を評価する実験を行った．その結果, PyTorch の決定論的モード [82] を使って決定論的な状態遷移を行うこと, GPU メモリ上のデータを CPU メモリに移動すること, 大きなデータブロックのハッシュ化の 3 点が学習時間に影響が与えることが分かった．最後の 2 点の影響は, 処理の並列化によって軽減可能である．また, 紛争解決プロトコルを実行する際に必要なオーバーヘッドについて, 第 4 章のチェーン成長振舞い分析を用いて評価した．以上より, 本手法が実際のアプリケーションに適用可能であることを確認し, 紛争解決プロトコルが実行される最大期待待ち時間を考慮した上でシステムを設計することが可能になった．

8.2 本研究の前提と目的

本節では, 状態遷移マシンとして見たブロックチェーンシステムと, 最も普遍的な深層学習手法である BP (Back-propagation) [83] について説明する．これらを踏まえ, 本研究の目的を明らかにする．

8.2.1 状態遷移マシンとしてのブロックチェーンシステム

ブロックチェーンは, S. Nakamoto がディジタル通貨である Bitcoin [1] のために導入した技術である．Bitcoin では, 個々のディジタル通貨とその所有権に関する状態遷移に

ついて, 合意形成を行っている. 例えば, 5 Bitcoin しか所有していないユーザから 10 Bitcoin を送金するトランザクションは, 無効と見做すようにしている. ディジタル通貨の状態遷移に特化した Bitcoin を一般化した Ethereum [3] では, チューリング完全なプログラミング言語で記述可能な任意の状態遷移について合意形成可能である. Ethereum における, 状態遷移は以下の式で表現される.

$$\sigma_{t+1} = \Upsilon(\sigma_t, \mathrm{Tx}). \tag{8.1}$$

式 8.1 は, トランザクション Tx で定義される状態遷移関数 Υ によって, σ_t から σ_{t+1} へブロックチェーンシステムの状態が遷移することを意味する. 上式に基づき, Namecoin [61] のようなドメインネームシステムなど, これまでは信頼できる第三者を通じて実現できたアプリケーションをブロックチェーン上で再構築可能となった [64]. 前節で述べたように, ニューラルネットワークの学習プロセスは, ネットワークパラメータが決定論的に更新されれば, 状態遷移として表現することができる. このため, ブロックチェーン上で深層学習のプロセスを原理的に検証可能である. しかし, ブロックチェーンのスケーラビリティの問題 [11, 27] から, 現実的には困難であるため, オフチェーン技術である (ORU: Optimistic RollUp) を利用する. ORU のプロトコル動作の仕組みは, 第 2 章 で説明した.

8.2.2 深層学習 (Deep Learning)

深層学習では, DNN (Deep Neural Network) の重みやバイアスといったパラメータを更新する. 典型的な DNN は, 入力層, 複数の隠れ層, 出力層から構成される. 各隠れ層は一定数のニューロンを持ち, 各ニューロンは複数の入力と 1 つの出力を持つ. 全結合層 (Fully connected layer) の場合, 前の層の全てのニューロンの出力が, 各ニューロンの入力として接続される. 2 つのニューロン間の接続には重みが割り当てられており, 各ニューロンにはバイアスがある. これら重みとバイアスは, モデルパラメータとして定義され, 学習中に更新される.

誤差逆伝搬法 (BP: Back Propagation) [83] は, 深層学習において最も普遍的な学習方法である. BP 学習アルゴリズムでは, 以下 3 つの手続きを繰り返し実行する.

- 順伝搬
- 逆伝搬
- 重み更新

この 3 つの手続きがどのように行われるか, 全ての層が全結合である例を用いて説明する. 順伝搬処理は, 式 8.2, 8.3 を全ての層に対して順次計算することで完了する.

$$U^{(l)} = W^{(l)} Z^{(l-1)} + \boldsymbol{b}^{(l)} \mathbf{1}_N^T, \tag{8.2}$$

$$Z^{(l)} = f^{(l)}(U^{(l)}). \tag{8.3}$$

$l(= 1, 2, ...)$ は層のインデックス, $W^{(l)}$ は層 l と $l-1$ 間の重み行列, $bmb^{(l)}$ は層 l におけるバイアスベクトル, $\mathbf{1}_N$ はサイズ N で全てが 1 のベクトル, N は学習時のバッチサイズである. $Z^{(l)}$ は, 各列が学習サンプルの一つに対応する層 l の活性化ベクトルを表す行列である[1]. $f^{(l)}(\cdot)$ は層 l の関数で, 入力する行列の各要素に活性化関数を適用した同サイズの行列を返す関数である. 順伝搬計算が終了すると, 最終層のニューロンの出力が得られ, DNN の目的関数を計算することができる.

誤差逆伝搬処理は, 式 8.4 ∼ 8.6 を, 順伝搬の時の逆順に, 全層に対して計算することで完了する.

$$\Delta^{(l)} = f^{(l)\prime}(U^{(l)}) \otimes \left((W^{(l+1)})^T \Delta^{(l+1)}\right) \tag{8.4}$$

$$\partial W^{(l)} = \frac{1}{N} \Delta^{(l)} (Z^{(l-1)})^T \tag{8.5}$$

$$\partial \boldsymbol{b}^{(l)} = \frac{1}{N} \Delta^{(l)} \mathbf{1}_N \tag{8.6}$$

$\otimes$ はアダマール積, $\Delta^{(l)}$ は勾配行列で, 各要素は活性化前の層 l の対応するニューロンの出力を変数とした目的関数の微分係数である. $\partial W^{(l)}$ は層 l と層 $l-1$ 間の重みの勾配行列, $\partial \boldsymbol{b}^{(l)}$ は層 l でのバイアスの勾配ベクトルである. 誤差逆伝搬処理では, 微分係

[1] $Z^{(0)}$ は, 入力として深層ニューラルネットワークに与えられるバッチサイズ N のサンプルデータである.

数を得るために, 既に式 8.2, 8.3 で計算されている活性化前後のニューロンの出力を用いる. つまり, この計算には DNN の部分的な状態が必要となる.

モデルパラメータの更新処理では, 式 8.7, 8.8 により重みとバイアスが更新される.

$$W^{(l)} \leftarrow W^{(l)} - \epsilon \partial W^{(l)} \tag{8.7}$$

$$\boldsymbol{b}^{(l)} \leftarrow \boldsymbol{b}^{(l)} - \epsilon \partial \boldsymbol{b}^{(l)} \tag{8.8}$$

ϵ は学習率である. これら手続きは確率的勾配降下法とも呼ばれる. 以上のように機械化された計算から, 与えられた訓練データに対して決定論的な BP アルゴリズムを実行することで, 深層学習をエミュレートすることができる.

8.2.3 本研究の貢献

本研究では, ブロックチェーン上で実行される紛争解決プロトコルに基づき, DNN モデル生成プロセスの検証を可能とする手法を提案した. 深層学習過程で行われる計算では, 巨大な行列演算の実行や, ネットワークの中間状態を保存するための大容量メモリが必要である. そこで, 通常の計算を考慮する Arbitrum [15] や TrueBit [39] とは異なる紛争解決プロトコルを提案する. 本プロトコルでは, モデル検証者が DNN のネットワークパラメータの中間状態を提供することで, モデル提供者は初期状態から深層学習を行わなくてよい. その代わりとして, 紛争解決プロトコルを実施するために必要なステートルートを部分的に再計算する必要がある.

複数の DNN に対して, 深層学習の実行と全ステートルートの計算にかかる実行時間を測定した. これにより, ブロックチェーン上に保存されるルートハッシュ計算が学習実行時間に与える悪影響について評価した. また, ブロックチェーン上で実行される紛争解決プロトコルの最大期待実行時間について, 第 4 章のブロックチェーン成長の振舞いの分析を用いて評価した.

8.3 提案: DNN の状態遷移検証手法

本節では, DNN の状態遷移を検証するための証明を提案し, その動作原理について説明する.

8.3.1 役割

提案システムでは, 4 種類の役割が存在する.

1) *Verifier*:

Validator は, ブロックチェーンの管理者である. 本システムでは, DNN の全状態遷移を定義するルートハッシュ (図. 8.3 のグローバルマークルルート) と全学習データのハッシュをセキュアに保存するために用いられる. また, 紛争解決プロトコルを実行する一連のトランザクションもブロックチェーン上で処理される.

2) *Model provider*:

Model provider は, DNN モデルを学習させ公開するユーザーである. モデル提供者は, 全状態遷移を定義するルートハッシュと, 全学習データのハッシュをブロックチェーン上に保存する. また, 全学習データと各エポックプロセスを検証するためのハッシュ (図. 8.3 を参照) を, ブロックチェーン外部のストレージに保存する (これらデータは, 可用性を高めるために IPFS (Inter-Planetary File System) [84] などを用いる). 学習データとエポックプロセスを検証するためのハッシュの完全性は, ブロックチェーンのハッシュにより保証される. 仮に, モデル検証者 (Validators) がモデル提供者のモデルが正しく生成されていないと主張した場合, モデル提供者は 8.3.4 で説明する紛争解決プロトコルを通じて, 応答する必要がある.

3) *Model validator*:

Model validator は, 与えられた DNN モデルが正しく生成されていないことを, 紛争解決プロトコルを通じて, 主張する. この紛争解決プロトコルの結果, 正しい状

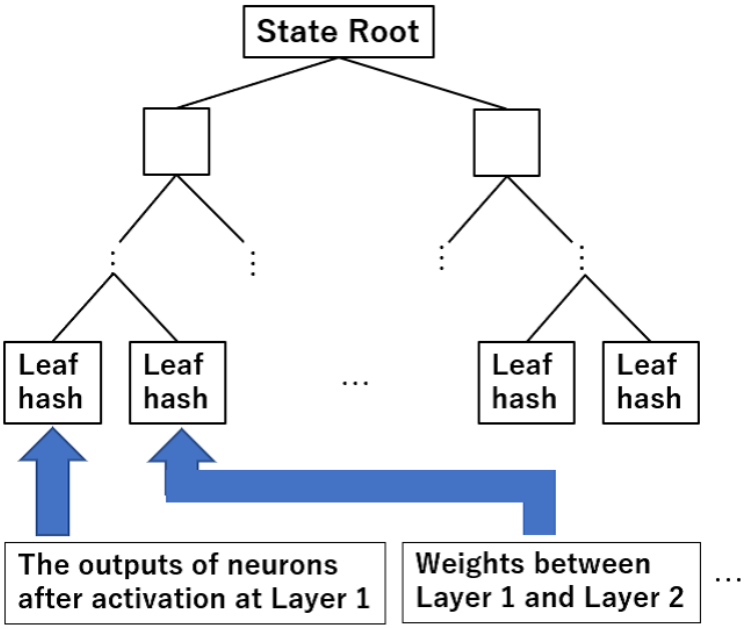


図 8.1: データブロックは, 各レイヤーの計算を独立に検証するために必要な最小の大きさとする. データブロックのハッシュをリーフとするマークル木により, ステートルートは計算される. 各レイヤーの計算が終了し, データブロックの値が変化する度に, ステートルートは更新される

態遷移が行われていない箇所をブロックチェーン上で主張することができる. 指定された該当箇所の計算をエミュレートするためのデータは, モデル検証者によってブロックチェーン外のストレージ (例えば, IPFS) に保存され, その状態遷移が正しさはライトクライアントによって判定される.

4) *Light client*:

Light client は, 紛争解決プロトコルで得られた正しい状態遷移が行われていないと主張される箇所をエミュレートする. モデル検証者により与えられた DNN の中間状態のデータを用いて, 1 層分のみの計算をエミュレートするため, 深層学習のための十分な計算資源 (例えば, 時間や空間資源) は要求されない.

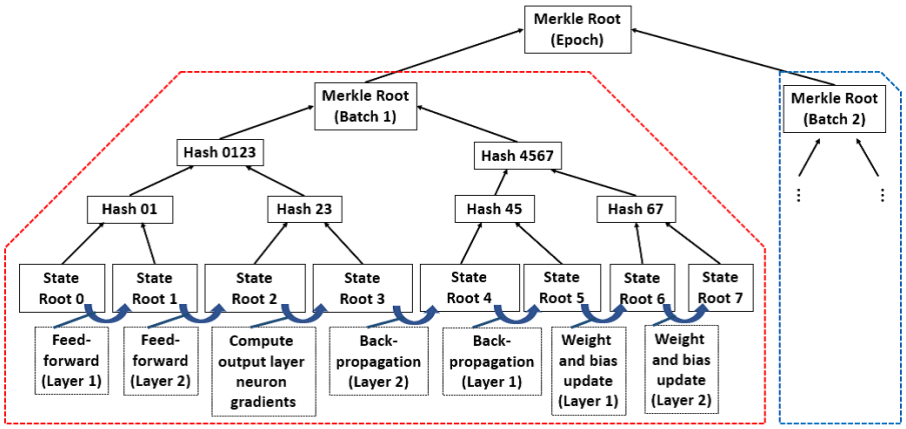


図 8.2: この図は, 入力層 (第 0 層), 隠れ層 (第 1 層), 出力層 (第 2 層) からなる 3 層ネットワークのステートルートを計算する例を示している. ステートルートは, DNN モデルのすべての内部状態 (例えば各層のニューロンの出力や, レイヤー間のバイアス, 重み) を一意に決定する

8.3.2 ステートルートとグローバルマークルルートを用いた紛争解決

計算リソースに制約のあるライトクライアントが, モデル検証者によって不正と主張された状態遷移箇所をエミュレートできるように, まず DNN の学習プロセス全体を 8.2.2 で説明したレイヤーベースの計算へと分解する. また, 各レイヤーベースの計算をエミュレーションする際に用いられるデータの完全性 (Integrity) を保証するためにステートルートを定義する.

ステートルートは, 各レイヤーベースの計算を検証するために必要なデータブロックのハッシュをリーフハッシュとしたマークル木のマークルルートとして定義される (図 8.1 を参照). 各データブロックの完全性は, そのリーフハッシュに対応したマークル証明によって保証される. そのため, 関係するデータブロックとマークル証明さえ与えられれば, 対象となるレイヤーベースの計算をエミュレート可能である.

学習プロセスの実行過程で, レイヤーベースの計算が実行される度にデータブロックの値が更新されていく. この結果, ステートルートも次々と値が更新される. 第 2 章で

解説したように, 隣接した二つのステートルートと, その状態遷移を計算するのに必要なデータブロック, 及びそのマークル証明が与えられれば, その区間での状態遷移の正しさを検証することができる. 各ステートルート群の完全性を保証するために, それらステートルートをルートハッシュに持つマークルルートを, エポックごとに計算する. 図 8.2 にその様子を示す (この例の場合, 深層学習のバッチサイズは 2 であり, ステートルートは $8 \times 2 = 16$ 存在している). この結果, 図 8.2 のマークルルート (エポック) の完全性が保証されれば, リーフハッシュであるステートルートの完全性をマークル証明によって保証することができる.

図 8.3 の赤点線内部のマークルルート (エポック) の完全性は, マークルルート (エポック) をリーフハッシュに持つグローバルマークルルートによって保証される. グローバルマークルルートは, モデル作成者によって, ブロックチェーン上に保存される. 以上より, グローバルマークルルートを起点とするマークル証明を再帰的に実行することで, モデル提供者は自身が行った学習プロセスの状態遷移を定義する全てのステートルートの完全性を提示可能となる.

8.3.3 ライトクライアントによるレイヤーベースの計算の検証

深層学習のレイヤーベースの計算は, 8.2.2 で説明したように行われる. 各計算を行うために必要なデータブロックは, 学習中の DNN の全データのごく一部である. 例えば, ニューロンの出力は, 前層のニューロンの出力, 層間の重み, 現在の層のバイアスを用いて, 式 (8-2, 3) のように計算することができる. 各データブロックの完全性は, 図 8.1 に示すステートルートと対応するマークル証明によって保証される. 全ステートルートの完全性も, 図 8.3 に示すブロックチェーン上のグローバルマークルルートと, 再帰的なマークル証明によって保証される.

ライトクライアントは, 第 2 章で説明した (ORU: Optimistic RollUp) と同様に, 隣接する 2 つのステートルート間での状態遷移を検証する. 検証対象となっているレイヤーベースの計算の後, 一部のデータブロックが更新され, ステートルートも更新される. 更

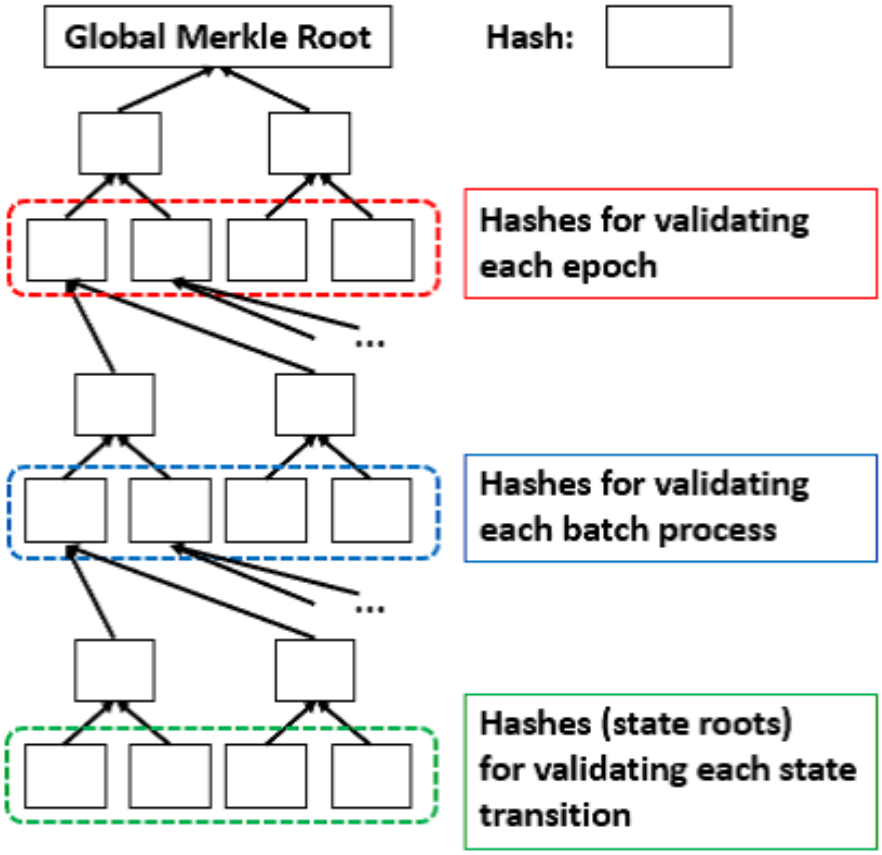


図 8.3: 緑色の破線で示したステートルート (リーフハッシュ) と中間ノードのハッシュを順次計算することで得られるグローバルマークルルート

新対象のデータブロックから新たなステートルートを計算するため, 一つ前のステートルートに関する, 更新されるデータブロックのマークル証明を予め与えておく. この結果, 更新されたデータブロックのハッシュ値と, 中間ノードのハッシュ値を用いて, 新たなステートルートを再計算可能となる.

8.3.4 紛争解決プロトコル

計算能力に制限のあるライトクライアントに納得可能な紛争解決を行うためには, ライトクライアントでも検証可能な不正な状態遷移部分が提示されればよい. 第 2 章で説明したように, これは誤った状態遷移が発生する隣接する 2 つのステートルートを指定することである. 隣接する 2 つのステートルートがあれば, 直前のステートルートから DNN の状態を再構築し, 再計算された次のステートルートが, ブロックチェーン上のグローバルマークルルートによって完全性が保証されたステートルートと一致するかどうかを確認できる.

隣接する 2 つのステートルートを得るため, Arbitrum [15] でも採用されている対話型 2 分法プロトコルを採用する. このプロトコルでは, モデル検証者が図 8.3 の赤の破線内で示す各エポックの検証用マークルルート (エポック) を, 深層学習の実行過程で逐次的に計算し, IPFS などのオフチェーンストレージ上にモデル提供者によって保存された値と完全に一致するかを確認する. 仮に, 不一致のマークルルート (エポック) を発見した場合, アルゴリズム 1 に示す対話型 2 分法プロトコルを実行する.

本システムでは, DNN モデル提供者は全てのステートルート (エポック) を保持している. そのため, モデル提供者の図 8.3 の赤線内のマークルルート (エポック) より, 上部のマークル木を容易に再構築できる. モデル検証者は, モデル提供者に対しアルゴリズム 1 の対話型 2 分法プロトコルを実行することで, 自身の再検証の結果得られたマークル木と不一致となる最も左に存在するマークルルート (エポック) を確認できる (仮に, モデル提供者が対話型 2 分法プロトコルに有限時間内で応答しない場合, 与えられたモデルは不正であると判定される).

その後, 不正な状態遷移があると主張されているエポックについて, モデル検証者がその段階までの学習パラメータを IPFS などを通じて, モデル提供者に通知する. このパラメータを用いることで, モデル提供者は対象となるエポックのステートルートを再計算できる. この再計算されたステートルート群を用いて, 再度アルゴリズム 1 に示される対話型 2 分法プロトコルを実行可能である. この結果, モデル検証者とモデル提供者が構築した 2 つのマークル木で, 最も左に存在する不一致となるステートルートを取得することができる. これがモデル検証者が, 不正な状態遷移が行われていると指摘できる隣接する 2 つのステートルートの指定方法である.

これらの対話的な手続きは, ブロックチェーン上の一連のトランザクションによって行われるため, 第三者により容易に監査可能である. また, 対話型 2 分法プロトコル実行時のタイムアウト時間の設定に, マイナーによる操作が困難であるブロック番号を利用した時間依存のロジックを用いる. この時間依存ロジックで指定されたブロック番号に到達するまでの時間は, 第 4 章で解説したチェーン成長の振舞い分析によって評価可能である.

以上の手続きで, ライトクライアントは再検証すべきレイヤーベースの計算を定義した隣接する 2 つのステートルートに関する情報が得られる. ライトクライアントがレイヤーベースの計算のエミュレートに必要とするデータブロックは, モデル検証者により IPFS などのブロックチェーンの外側のデータベースに保存する. その完全性はブロックチェーン上に保存したグローバルマークルルートと再帰的マークル証明によって保証される.

モデル提供者または, モデル検証者による悪意ある行動を減らすために, Arbitrum [15] で用いられているペナルティの仕組みを導入する. この仕組みでは, モデル提供者とモデル検証者が対話型 2 分法プロトコルに参加するために, ブロックチェーン上に自分のお金を預ける必要がある. 判定者の検証によって, 不正が発覚した側は, この預金 (デポジット) を失うことになる. 対話型 2 分法プロトコルが終了せず, モデル提供者が常にプロトコルに応答する場合, モデル検証者はデポジットを失うというペナルティを受ける. それ以外の場合は, モデル提供者がペナルティを受ける. これらの判定結果は, 判定者によっ

表 8.1: 提案した検証可能な深層学習と, 従来の深層学習とのエポックあたりの実行時間の比較. FCN は入力層と 8 つの全結合層から構成される. CNN は, 入力層, 8 つの畳み込み層, 2 つの全結合層から構成される

Network	**Batch Size**	**Normal: Epoch Time (StdDev.)**	**Verifiable: Epoch Time (StdDev.)**	**Verifiable: (Reduce Transferred Neurons) Epoch Time (StdDev.)**
FCN	100	11.219 [sec] (0.129)	20.313 [sec] (0.078)	19.536 [sec] (0.060)
FCN	200	8.564 [sec] (0.100)	13.449 [sec] (0.098)	12.939 [sec] (0.080)
CNN	100	14.317 [sec] (0.129)	57.369 [sec] (0.244)	40.327 [sec] (0.548)
CNN	200	12.664 [sec] (0.225)	48.586 [sec] (0.223)	31.846 [sec] (0.170)

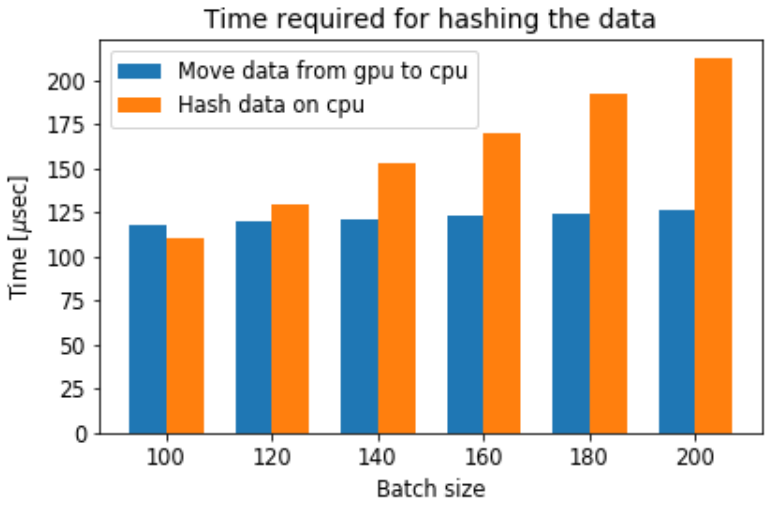
てブロックチェーン上に公開される. また, 判定者は, 計算資源に制限のあるライトクライアントによって監査される.

8.4 概念実証

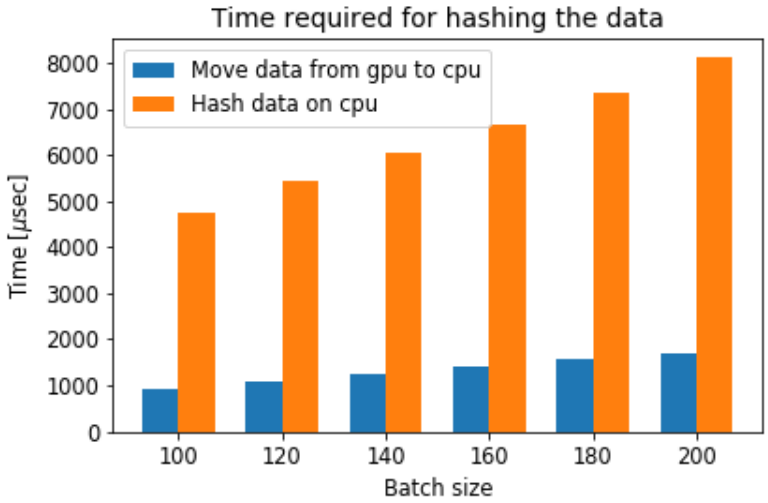
本概念実証では, 複数の学習モデルに対して, 提案した検証可能な深層学習の実行時間を評価するために, ケーススタディを行う. 提案したシステムでは, 決定論的に定まるステートルートを得るために, ニューロン, ウェイト, バイアスに関するデータブロックを GPU メモリから CPU メモリに同期して移動させ, その暗号学的なハッシュを得るようにしている. 本節では, この手続きが深層学習の実行速度に与える影響を定量的に評価

Algorithm 1 モデル提供者のハッシュと最初に不一致となった最も左のリーフノードのハッシュを取得する対話型 2 分法プロトコル

```
// Validator sets root node about target epoch.
tgtNode = rootNode // Given from offchain storage
for iter = 1, 2, ..., merkleTreeHeight do
    // Validator's request phase
    if timeout() then Validator's assertion is invalid.
    Validator.requestChildNodeData(tgtNode)
    // Provider's respond phase
    if timeout() then Provider's model is not reliable.
    Provider.sendChildNodeData(tgtNode)
    // Validator's check and 'tgtNode' update phase
    leftNode, rightNode =
        Validator.receiveChildNodeData(tgtNode)
    if hash(leftNode.hash, rightNode.hash)
                                != tgtNode.hash then
        Provider's model is not reliable.
    else
        if leftNode.hash != localLeftNode.hash) then
            tgtNode = leftNode
        else tgtNode = rightNode
return tgtNode /* This node is a leaf node. */
```



(a) 8.4.2 で定義した FCN の全結合層の出力ニューロンをハッシュするのにかかる時間



(b) 8.4.2 で定義した CNN の畳み込み層の出力ニューロンをハッシュするのにかかる時間

図 8.4: (a) と (b) はデータブロックのハッシュ化に必要な時間である．青いバーは, GPU から CPU へのデータ転送時間であり, オレンジ色のバーは CPU 上でデータのハッシュ化に必要な時間である

する.

8.4.1 実験環境と各種設定について

エポックあたりの実行時間を評価するため, 1 台の PC と, 深層学習フレームワーク PyTorch を用いて, いくつかのニューラルネットワークに対して深層学習を実施した. PC の OS は Windows 10 Home (64 ビット) で, クアッドコアのデスクトップ CPU (Intel i7-6700 CPU @ 3.40GHz), 32 GB RAM, GPU カード (NVIDIA GTX 1060) を 2 枚搭載している. CPU のコア数は 4 であり, スレッド数 (論理プロセッサ数) は 8 である. 深層学習及び, ステートルートを計算するプログラムの実装時に, Python 3.6.5 と PyTorch 1.1.0 を使用した. DNN 内の各変数は PyTorch 上で実装された 32 ビット浮動小数点として定義されている. デフォルトの暗号学的ハッシュ関数として SHA-256 を採用した. また, 本実験では 500MB 以上の大きなデータをハッシュ化する際には, SHA-512 を使用した. これは, 64bit マシンにおいて, 巨大なデータのハッシュを計算する際に, SHA-512 の方が SHA-256 より高速に動作するためである [85]. また, 500 MB 以上のデータは 8 ブロックに分割され, 8 個の論理 CPU 上で並列にハッシュ化し, それら 8 つのハッシュ値を連結したハッシュを最終的なハッシュとした. この結果, 大容量データのハッシュ化にかかる実行時間は短縮される. なお, 使用した SHA-256 と SHA-512 は, いずれも Python 3 の標準ライブラリの実装を利用した. GPU と CPU メモリ間の転送速度を増加させるため, PyTorch [86] のピンメモリを使用した.

8.4.2 DNN の構成

本実験では FCN, CNN, U-Net の合計 3 つの DNN を用いた. FCN (Fully Connected Network) は入力層と 8 つの全結合層からなり, 出力層を除く全結合層は活性化関数として ReLU (Rectified Linear Unit) を用いている. 各全結合層のニューロン数は 200 であり, 出力層のニューロン数は 10 である.

表 8.2: 提案した検証可能な深層学習と, 従来の深層学習のエポックあたりの実行時間の比較. U-Net のネットワーク構造については 8.4.2 で説明している

Network	**Batch Size**	**Normal: Epoch Time (StdDev.)**	**Verifiable: Epoch Time (StdDev.)**	**Verifiable: (Reduce Transferred Neurons) Epoch Time (StdDev.)**
U-Net	10	141.045 [sec] (0.692)	411.756 [sec] (1.644)	292.089 [sec] (1.060)
U-Net	20	130.659 [sec] (1.022)	370.184 [sec] (2.573)	256.080 [sec] (1.041)

CNN (Convolutional Neural Network) は, 入力層, 8 つの畳み込み層, 2 つの全結合層からなり, 出力層を除く各畳み込み層と全結合層は活性化関数として ReLU を使用している. 畳み込み層では, パディングサイズを 1 とし, ストライド 1 の 3×3 の畳み込みを使用した. 各畳み込み層のチャンネル数は 32 であり, 2 つある全結合層の最初の出力ニューロン数は 128 であり, 後ろの出力ニューロン数は 10 である. 最後の畳み込み層のニューロンの出力は, 最初の全結合層に接続するために平坦化されている.

上記の DNN に加えて, より実用的なタスクに対する性能を評価するために, 医療用セグメンテーションタスクに使用される U-Net [87] を本概念実証で用いる. 8.4.1 と 8.4.3 で説明している実験環境と学習データに合わせて, 以下の点で U-Net を修正した. ダウンサンプリング, アップコンバージョンの各ステップにおいて, パディングサイズ 1, ストライド 1 の 3×3 の畳み込みを 2 回行った (U-Net を提案した論文 [87] では, パディング無しの 3×3 の畳み込みを行っていた）. 各畳み込みを実行する際には, 活性化関数である ReLU を繰り返し使用した. 最初の畳み込み層のチャンネル数は 32 とし, ダウンサンプリング毎にチャンネル数を 2 倍にして 1024 になるまで行った. アップコンバートの各段階では, 32 チャンネルになるまでチャンネル数を半分にしていった. 出力層で

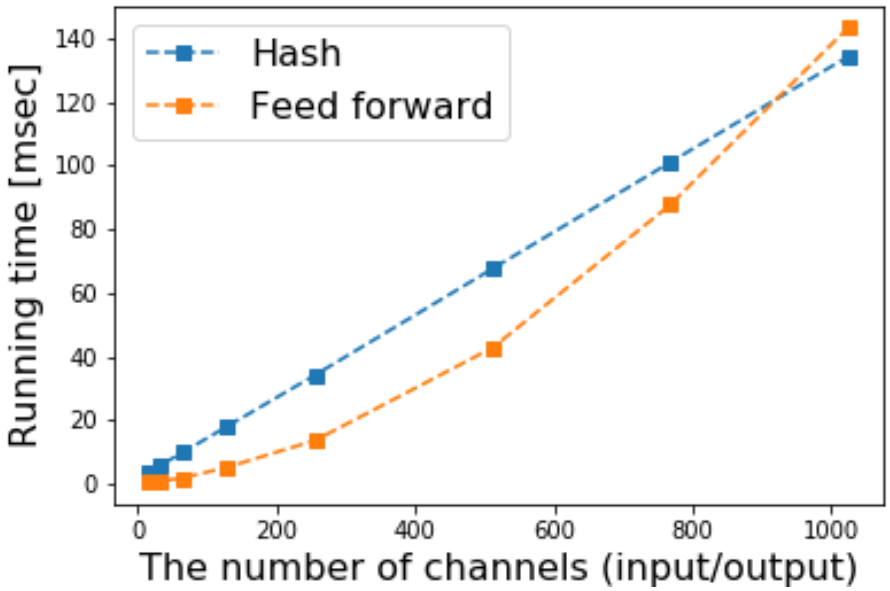


図 8.5: 1 つの畳み込み層に対する順伝搬処理と, 出力ニューロンのデータブロックをハッシュするのにかかる時間 (前後の層のチャンネル数は同時に変更)

は, 1×1 の畳み込みを行った後に, シグモイド関数による活性化を行った. チャンネル数は 8.4.3 の学習データに合わせるために 1 にした.

8.4.3 学習データと手続き

FCN と CNN は手書きの数字分類タスクを学習する. FCN と CNN の学習には, 手書き数字の MNIST データベース [88] から得た 6 万サンプルの学習データセットを用いた. FCN では, 28×28 画像データを平坦化した上で DNN への入力とした. 出力層の 10 ニューロンを用いて, 両 DNN の目的関数としてクロスエントロピーを使用した. 表 8.1 の平均エポックタイムを算出するため, エポック回数は 100 回とした.

U-Net では, セグメンテーションタスクを学習する. このために, 魚類認識のグランドトゥルースデータ [89] を利用した. 学習データの魚種は, Plectroglyphidodon dickii で

ある．学習データは 2,683 個であり, それをデータ拡張により 6,000 個 に増やした．また, 入力画像を 224×224 ピクセルにリサイズし, 各ピクセル値を -1.0 から 1.0 に正規化した．マスク画像の値も 0.0 から 1.0 に正規化した．U-Net の目的関数としてダイス係数を用いた．表 8.2 の平均エポックタイムを計算するためのエポック数は 40 とした．3 つの DNN は全て PyTorch の確率的勾配降下の実装で学習させた．

8.4.4 検証可能な DNN 学習

8.3.4 で説明した紛争解決プロトコルを可能にするためには, 学習中のステートルートを決定論的に計算する必要がある．このために, PyTorch の決定論的モード [82] を利用した．各レイヤーベースの計算の後, 更新されたデータブロックは GPU から CPU に転送され, 新しいステートルートを計算するために暗号学的ハッシュ関数でハッシュ化される．この際, データブロックは GPU から CPU への転送が終わるまで上書きされてはならない．このために, GPU 上の各処理と CPU 上の更新されたデータブロックのハッシュ化処理を同期しながら実行した．具体的には, GPU 上で更新されたデータブロックは, すぐに CPU に転送され, ハッシュ化される．その後, 8.2.2 で説明した順伝搬や逆伝搬といった GPU 上での処理が終了すると, GPU は CPU 上で進行中のハッシュ計算が終了するのを待つようにした．

また, 大規模データのハッシュ化にかかる時間を短縮するため, 一部のレイヤーの出力ニューロンデータブロックをハッシュ化しない深層学習も実施しました．FCN と CNN の場合, 1 層おきに出力ニューロンのデータブロックを無視した．このようにした結果, 紛争解決プロトコルの後に検証を行う判定者は, 2 つ分のレイヤーベースの計算を行う必要が出る (データブロックとして存在しない層の出力の完全性は, ステートルートで保証されないため)．U-Net の場合, 二つの畳み込み層に単純に挟まれている (合流などがない) 部分のデータブロックのハッシュ化を行わないことで, データブロック数を 23 個から 11 個に減らした．

8.4.5 実験結果

表 8.1 は, 8.4.2, 8.4.3 の設定で FCN と CNN を学習させた場合の平均エポックタイムである. 図 8.4 の (a) は, FCN の全結合層で 200 出力ニューロンのデータブロックをハッシュ化するのに要した時間である. 図 8.4 の (b) は, チャンネル数が 32 の CNN の各畳み込み層の出力ニューロンのデータブロックをハッシュ化するのに要した時間である.

表 8.2 は, 8.4.2, 8.4.3 で説明した U-Net の学習に要した平均エポック時間である. 図 8.5 は, 直前の層と現在の層のチャンネル数を同じにした場合の, 現在の層に対する順伝搬処理の実行時間と, 出力ニューロンに関するデータブロックのハッシュに要する時間を比較したものである. 層への入力ニューロンデータは, バッチサイズ 100 で固定し, 28×28 の画像であり, この層ではパディングサイズ 1, ストライド 1 の 3×3 の畳み込みとした.

8.5 議論

8.5.1 提案した検証可能な DNN と既存研究との比較

Practical delegation of computation [90] も提案した手法と同様に, あるタスクの計算結果の正しさを証明するために, ステートルートによる紛争解決プロトコルと似た仕組みを採用している. この手法の場合, 最悪の場合, 委任された検証者は対象タスクを 2 回実行する必要がある. このオーバーヘッドを減らすために, [90] の著者らは, 計算途中の状態を随時保存するようにすれば, 途中の状態に到達するまでの実行時間を短縮できると述べている. 提案手法では, 深層学習をゼロから実行することを避けるために, モデル検証者がエポックごとに DNN モデルのパラメータを保存している. ニューラルネットワーク内のバッチレベルでの出力ニューロンといったデータと比較して, DNN モデルのパラメータのデータサイズは大きくないため, そのコストは比較的大きくない.

Slalom [91] のようにネットワークのパラメータを量子化することで, 検証可能な推論

用 DNN も提案されている．順伝搬と逆伝搬の計算は原理的に同じであるため, Slalom を DNN の学習に応用できるとも主張されているが, 量子化した DNN の学習の実行が困難であるという問題もある．これは, 量子化された重みの変化が学習精度を大きく劣化させてしまうためである．一方, 本手法では, 決定論的な浮動小数点演算さえ行えば通常の DNN と同様な学習を行える．このため, 本手法は浮動小数点数で学習する他の DNN にも容易に適用可能という利点がある．

本手法では, 学習データのプライバシーを考慮していない． 検証可能な推論を行う Slalom [91] や SafetyNets [92] では, 入力データのプライバシーを考慮している．しかし, これらの手法は DNN の学習データのプライバシーを保証する代わりに, 透明性を失わせる．汚染された学習データを利用するバックドア攻撃 [78, 79] への対処という点では, 単に学習データのプライバシーを保証することは欠点となる．この問題は, 学習データを提供した参加者が高確率で正しいデータを提供するようにするインセンティブ機構 [93] によって緩和することができるとされている．しかし, 提案手法では学習プロセス全体を完全に監査するため, この内容は本研究の範囲外である．

8.5.2 提案した検証可能な DNN 学習が持つ特徴

表 8.1 は, 検証可能な深層学習を実行する時間が, FCN で約 1.81 倍, CNN で 4.01 倍遅くなっていることを示している (バッチサイズは 100 の場合). FCN と CNN も, バッチサイズが大きくなるにつれて検証可能な深層学習の実行時間が相対的に短縮されている (バッチサイズを 200 とした場合, 検証可能な深層学習の実行時間は FCN の場合 1.57 倍, CNN の場合は 3.84 倍遅くなる). これは, 図 4 のデータブロックをハッシュする全手続きの実行時間に対する, GPU から CPU へのデータ転送開始するためのオーバーヘッドの割合が減少したためと考えられる．

図 5 は, 対象となる畳み込み層とその前の畳み込み層のチャンネル数が同じように増加した場合, 順伝搬処理の時間は二次関数的に増加することを示している．一方, 対象の層に関する出力ニューロンのデータブロックをハッシュ化する時間は, 線形に増加して

いる．このため, チャンネル数の多い畳み込み層を持つネットワークでは, 提案した検証可能な深層学習の実行時間が相対期に早くなる．例えば, 表 8.2 に示すように, チャンネル数が非常に多い U-Net では約 2.92 倍遅くなっている（バッチサイズは 10）．一方, 表 8.1 の CNN は, チャンネル数が少ないため, 約 4.01 倍遅くなっていることがわかる (バッチサイズは 100).

8.5.3 提案した検証可能な深層学習の限界

検証者が深層学習を実行するために必要なデータの可用性については, まだ考慮していない．検証者が全深層学習プロセスを実行するためには, 学習データセットを全て取得する必要がある．ブロックチェーンの外側のデータの可用性を保証するために, Erasure Coding [37] などの符号化方式を用いて, データを冗長化する手法が存在する．今後, 本提案にこの手法を提案することを検討している．

提案した検証可能な深層学習の実行時間は, 学習環境によって大きく異なる可能性がある．例えば, GPU と CPU 間の転送帯域が低いと, CPU 上で暗号学的ハッシュ関数の実行が遅延する．本論文では, GPU から CPU へのデータ転送や CPU でのハッシュ演算のスケジューリングに関する最適化については触れていない．一般的な最適化手法については, 今後の研究で検討する予定である．

8.5.4 紛争解決プロトコル実施時のパラメータ

紛争解決プロトコルにおいて実行される、アルゴリズム 1 対話型 2 分法プロトコルでは、節 4.1 で述べた、ブロック高 m 分経過を期限とする有効期限 (Expiration time) 付きトランザクションを使用する。また、2 分法プロトコルを実行するツリーの木の高さを H とした場合, $2H+1$ 回のトランザクション送信が必要である (2.4.1 を参照). 従って、紛争解決プロトコル終了までの期待時間は、平均チェーン成長時間の上限を T_{ub} とした場合、$m \cdot T_{ub} \cdot 2H$ となる。

つまり、ブロック生成速度 $\alpha = 1/15$, フォーク確率 $P_F = 0.01$, $m = 200$, $H = 10$ とすれば, 式 4.11 で計算される平均チェーン成長時間 T_{cg} の上限 T_{ub} を用いて, 紛争解決プロトコルの平均終了待ち時間 $E[W_{dr}]$ の上限を以下のように計算できる.

$$E[W_{dr}] \leq m \cdot T_{ub} \cdot 2H = 200 \cdot 15.78 \cdot 20 = 63120[\text{sec}] = 17.5[\text{hours}]. \tag{8.9}$$

また、今回のように、トランザクション送信前に DNN の学習などを必要とする場合、準備時間 $T_{prepare}$ が必要となる。このため、6.5.4 で行った、有効期限付きトランザクションが処理可能となる「トランザクション承認時間」の式は、以下のようになる。

$$T_{prepare} + T_{tx} < S_m. \tag{8.10}$$

S_m はチェーンが m ブロック分だけ成長する時間であり、T_{tx} は対象の有効期限付きトランザクション承認時間である。つまり、平均チェーン成長時間 T_{cg} の下限 T_{lb} を用いて、処理可能となる十分条件を次式で表現できる。

$$T_{prepare} + E[T_{tx}] < m \cdot T_{lb} = 200 \cdot 15.0 = 3000.0[\text{sec}] < E[S_m]. \tag{8.11}$$

8.6 関連研究

RDoC (Referred Delegation of Computation) は, 検証可能な計算を第三者機関に委任するためのプロトコルである [90, 94]. RDoC では, 資源に制約のあるクライアントが複数のサーバに計算を委任し, 各結果をクライアントに返す. 全ての結果が同じであれば, クライアントはその結果を受け入れる. それ以外の場合, 結果の異なるサーバー群に対して, 紛争解決プロトコルを実行し, 異なった計算を実行している箇所を検出する. 計算リソースに制約のあるライトクライアントは, 自身で計算できるほど十分に分割された計算をエミュレートすることで, どのサーバーが不正な計算を実行したか判断できる. Arbitrum [15] は同様な紛争解決プロトコルを採用しており, Arbitrum の One-step proof の検証にかかる時間と必要となるデータサイズを対数的に削減する手法を導入している.

Pinoccio [95] は公開型の証明 (Public proof) を用いることで, 検証可能な一般計算, 入力プライバシー, 非干渉性をサポートしている. しかし, このシステムは, 計算コストの高い準同型暗号を必要とする. このため, 計算を委任されたサーバーは自分の結果が正しいことを証明するために, 莫大な計算コストを要求される問題がある. 信頼されないクラウド上で検証可能な DNN 推論を行うシステムとしては, SafetyNets [92] が提案されている. SafetyNets を提案した著者らは, sum-check プロトコル [96] に基づく対話型証明を導入した. Slalom [91] は TEE (Trusted Execution Environment) と Freivalds のアルゴリズムを用いて, DNN 推論の安全な委任を提供する. 著者らは [92] で検討した設定において, Slalom が SafetyNets よりはるかに高速に DNN 推論を実行可能なことを示した. しかし, 量子化された DNN の学習は重みの値が大きく変化するため, 推論よりも困難である可能性が指摘されている [97].

8.7 結論

モデルの学習プロセスの透明性欠如により, DNN は社会的に信頼できないという問題がある. そこで, DNN のモデル生成過程を検証するための手法を提案した. 本手法では, モデル検証者の与えたモデルが正しく生成されていないことを, 計算リソースの少ないクライアントに対しても主張可能である. 学習プロセス全体はレイヤーベースの計算に分割され, 各計算の正しさは, 学習中の DNN の状態の完全性を保証するステートルートを用いることで検証される. ステートルートを計算するため, 次々と更新されるニューロンの出力やレイヤー間の重みなどの値を GPU と CPU 間で同期し, CPU 上で暗号学的ハッシュ関数を用いて計算する必要がある. そこで, 本提案でこの手続きが深層学習の実行時間に与える悪影響を評価するためのケーススタディを実施した. これより, 提案した検証可能な深層学習が実際のアプリケーションに適用可能であることが確認された.

第9章　結論

9.1　本論文の要約

合意形成により, 一貫性のある分散データベースを提供可能にするブロックチェーン技術への社会の期待は非常に高い. 例えば, 決済, スマートコントラクト, 所有権管理, ディジタル識別といった分野におけるブロックチェーン技術の採用率は高く, 銀行やサイバーセキュリティにおける広範な利用, 政府の活発な取り組みなどが行われている [9]. このため, ブロックチェーンインフラストラクチャー上でトランザクションを実行したいという潜在的需要は今後も増え続けることが予想される.

一方, ブロックチェーンには, システムに流入するトランザクション数が許容量を超えると, トランザクション承認遅延が発生するというスケーラビリティ問題 [10, 11] がある. トランザクション承認遅延は, チェーン成長の振舞い, 及びキューイング処理により生じ, これら 2 つの原因で発生する遅延は同時に改善できない (詳細は, 1.2.1 を参照). キューイング処理由来の遅延は, オフチェーン技術により改善可能である. ブロックチェーンの外側のデータベース (オフチェーンストレージ) を利用することで, トランザクションの送信回数やデータサイズが減少し, 実質的なブロックサイズを増加するためである (詳細は 1.2.2 を参照). しかし, 現状ではトランザクション承認遅延を劇的に改善させるまでには至っていない.

従って, トランザクションの処理内容が分散データベースに即時反映されないという前提の下, ブロックチェーンアプリケーションを開発する必要がある. 具体的には, データの参照, 有効期限のある手続き, 処理順序に意味のある手続きを行う際に留意しなければならない. 例えば, 意図した順序でトランザクションを処理するために, Namecoin [61]

や Lightning Network [14] では, ロック時間 (Lock time) 付きのトランザクションが利用されているが, トランザクション承認時間を予測できなければ適切なロック時間を設定できない. このため, 本論文では上記手続きを実行するブロックチェーンアプリケーションのスケーラビリティ, 性能, セキュリティに多大な影響を与えるトランザクション承認時間を, 数学的に評価する実用的な予測モデルの提案を行った. 本研究では, 第 4, 5 章において, 理論的かつ実用的な予測モデルを提案し, 第 6, 7, 8 章の実アプリケーションにおいて, それらモデルに基づいたパラメータ設定を行った上で, 性能, セキュリティ評価を行った. この結果, トランザクション承認遅延による性能, セキュリティへの影響を, パラメータ設定により制御する理論的な枠組みを提供した. 以下では, 本研究のより詳細な貢献について説明する.

9.1.1 本研究の貢献

第 4, 5 章では, トランザクション承認遅延の要因であるチェーン成長の振舞いと, キューイング処理について, 数学的な分析を行った.

ブロックチェーンの成長速度は, ネットワークを介するブロック伝搬遅延の影響で劣化する. 分散ノード間の同期ズレ補正のため, 合意形成の対象となるチェーンの成長速度に制限がかかるためである (3.1.1 を参照). 第 4 章において, ネットワーク遅延を特徴づけるブロック伝搬遅延の上限 D を用いず, 伝搬遅延と相関のあるフォーク確率 P_F (フォークの詳細なメカニズムは, 3.1.2 を参照) を用いて, PoW ブロックチェーンの平均チェーン成長時間 T_{cg} の上限 T_{ub} を求める手法を提案した. ブロック伝搬遅延から推定される D は, 測定に多数ノードを必要とする (論文 [40] では, 3,048 ノードを利用). 一方, P_F は, 単独ノードでも測定可能である.

フォーク確率を $P_F = 0.01$, ブロック生成速度を $\alpha = 1/600$ とした場合, 平均チェーン成長時間の上限は, 式 4.11 より $T_{ub} = 631.5$ [sec] のように与えられる. また, $P_F \to 0$ において, ブロック伝搬遅延が存在しない場合の平均チェーン成長時間 T_{cg} に, T_{ub} が漸近することを確認し, 提案モデルの妥当性を確認した.

また, ブロック伝搬遅延の上限 D が与えられている場合に, フォーク確率 P_F と併用することで, よりタイトな T_{cg} の上限と下限を導出できることを確認した (式 4.17, 式 4.24 を参照). これら上限と下限は, $D \to \infty$ を考えることで, D を仮定しないモデルに一致する ($P_F = 0.2$, $\alpha = 1/600$ とした場合の T_{cg} の下限は $T_{lb} = 600.0$ [sec] となる). これら結果より, 容易に観測可能なフォーク確率 P_F を用いて, チェーン成長速度の劣化を考慮した上で, ブロック高ベースのトランザクション有効期限や, ロック時間を設定できるようになった.

第 5 章では, メジャーなオフチェーン技術の一つである ORU に着目し, キューイング処理を待ち行列モデルにより記述し, ユーザーの待ち時間評価を行った. ORU では, ブロックチェーンの外側のデータベース (オフチェーンストレージ) を更新するオフチェーントランザクション (OTX) を一定周期でまとめ, ブロックチェーンのトランザクションとして送信する. この OTX が作成されてから, 一つのトランザクションとしてまとめられるまでの平均待ち時間 $\mathrm{E}[W]$ を評価した. オンチェーントランザクションの平均承認時間 $\mathrm{E}_{\mathrm{tx}}[T]$ (論文 [13] 等で導出されている) を用いれば, OTX の平均承認時間は $\mathrm{E}_{\mathrm{otx}}[T] = \mathrm{E}[W] + \mathrm{E}_{\mathrm{tx}}[T]$ と表せる.

OTX の到着率を $\lambda = 3.0$, OTX をまとめるバッチサイズを $b = 64$, タイムアウト時間を $T = 20$ [sec] とすれば, $\mathrm{E}[W] = 9.78$ [sec] となる (式 5.10 を参照). $b = 48$ とし, バッチサイズのみを減少させれば, $\mathrm{E}[W] = 7.82$ [sec] となり, 待ち時間は減少する. 一方, 仮定した待ち行列シナリオに基づいて, ORU システムからオンチェーンに放出されるオンチェーントランザクションのスループット TPS は増加する. 例えば, $\lambda = 3.0$, $b = 64$, $T = 20$ とした場合, $TPS = 0.051$ となる (式 5.16 を参照). バッチサイズのみ $b = 48$ とした場合, $TPS = 0.063$ となり, 土台となるブロックチェーンの負荷が増加したことが確認できる.

第 6, 7, 8 章では, 実アプリケーションを実装し, 第 4, 5 章のチェーン成長の振舞いと, キューイング処理に関する予測モデルをベースとして, システムのパラメータ設定を行った.

第 6 章では, ロック時間付きトランザクションを利用するブロックチェーンベースの ucode の所有権管理システムを提案した. 本システムでは, フロントランニング攻撃を緩和するために, 2 つのトランザクションを段階的に処理する ucode 割振りを行っている (詳細は, 第 6 章を参照). 後者のトランザクションにはロック時間が付いており, 前者のトランザクションが最長チェーンに含まれてから, m ブロック分だけチェーン成長するのを待つ必要がある. つまり, 第 4 章で導出した式 4.11 を利用し, ブロック生成速度 $\alpha = 1/15$, $P_F = 0.01$, $m = 20$ とすれば, ロック時間付きトランザクションを送信可能になるまでの時間の上限は, 315.6 [sec] と評価できる. また, フロントランニング攻撃が原理的に発生しない ucode 割振り手法を提案し, トランザクション承認の遅延時間を比較評価した.

第 7 章では, 第 6 章で提案した ucode 割振り手法を, オフチェーン技術を用いてスケールさせた. 割振り時の処理手続きは, 第 5 章で紹介した ORU と同じと見做せ (詳細は, 第 6 章を参照), 一定の ucode リクエストをまとめて, オンチェーントランザクションとして送信する. このため, 本事例はオンチェーン上で実装されたシステムをオフチェーン技術を用いてスケールする一例と考えられる. 本章では, オフチェーントランザクション (OTX) の承認時間の影響を評価 (式 5.10 を利用) した上で, まとめるバッチサイズ b, タイムアウト時間 T の設定を行った. ucode の割振りリクエスト到着率 $\lambda = 0.05$ (20 秒に 1 回), $b = 32$, $T = 600$ とすれば, 利用者の平均待ち時間は 286.73 [sec] となる.

第 8 章では, 有効期限付きトランザクションを利用する一例として, DNN (Deep Neural Network) モデル生成プロセスの検証をブロックチェーン上で可能にする手法を提案した. 本システムでは, 紛争解決プロトコルにタイムアウトを設けるため, 有効期限付きトランザクションを利用した (詳細は, 第 8 章を参照). 式 4.24 を利用して, 有効期限付きトランザクション作成にかかる時間 $T_{prepare}$ とトランザクションの平均承認時間を考慮した上で, ブロック高ベースの有効期限設定を行った. また, 上記設定に基づき, 式 4.11 を利用して, 紛争解決プロトコルの平均終了待ち時間の上限を評価した. ブロック生成速度 $\alpha = 1/15$, フォーク確率 $P_F = 0.01$, 有効期限を表すブロック高 $m = 200$, 紛

争解決プロトコルの反復回数 H とした場合, プロトコルの平均終了待ち時間の上限は, 63120[sec] $\approx$ 17.5[hours] と評価できる.

まとめると, 第 6, 8 章を通じて, ブロックチェーンの実アプリケーションが利用する有効期限付きトランザクションとロック時間付きトランザクションを, チェーン成長の振舞いの観点から分析し, 時間に関するパラメータを設定する事例を提供した. また, 第 7 章を通じて, オフチェーン技術である ORU と同等な手続きを経て OTX をまとめるシステムにおける, OTX の平均承認時間をキューイング処理の観点で評価し, バッチサイズ b やタイムアウト時間 T のパラメータを設定する事例を提供した. 以上から, 遅延を考慮した上でオンチェーン, 及びオフチェーンのトランザクションの振舞いを把握し, ロック時間や有効期限といったパラメータを数学的に決定することでトランザクションの振舞いを制御する実用的な枠組みが得られた. この結果, 数多くあるブロックチェーンインフラストラクチャーにおいて, トランザクション承認遅延を考慮したアプリケーション開発が容易になったと考えられる.

9.2 本研究の立ち位置と, 今後の展望

ブロックチェーンは, 2008 年に暗号理論に関するメーリングリストにて公開された S. Nakamoto の論文に端を発している. 2009 年 1 月には Bitcoin システムが実装され, その後も様々な暗号通貨が開発され, 一時は世界的な暗号通貨ブームとなった. ブロックチェーンは, 暗号通貨のみならず, 様々な価値交換サービスや分散台帳として世界中で利用されている.

特に, 2021 年以降, ブロックチェーンに啓発され, Web3 と呼ばれる新しい分散型のディジタル社会の世界観が盛んに議論 [98] されるようになっている. Web3 では, ブロックチェーンの仕組や機能を援用して, 社会における意思決定や価値交換を, 自律分散的に実現することが期待されている. これらのブロックチェーンにも, 以下のような様々な課題が指摘されている.

1. 技術課題

1.1. 機能要件

(1.1.a) スマートコントラクトを実装可能: ブロックチェーンの合意形成により一貫性の保証された分散データベースを, ユーザーの定義した任意のルールで更新できるようにすること (V. Buterin が提唱 [3])

(1.1.b) オラクル問題の解決, または緩和: ブロックチェーンの外部にある情報を, 信頼できる形式で分散データベースに保存すること ([99] を参照)

1.2. 非機能要件

(1.2.a) 性能: 単位時間に処理可能なトランザクション数 ([10, 11] を参照)

(1.2.b) スケーラビリティ: トランザクション承認遅延を増加させない, システムへのトランザクション到着率の臨界値 λ_{max} ([10, 11] を参照)

(1.2.c) セキュリティ: ビザンチンノードが一定の割合で存在する場合でも, 合意形成により一貫性のある分散データベースを実現可能か ([12] を参照)

(1.2.d) ブートストラッピングコスト: 各分散ノードが一からブロックチェーンの分散データベースを作成し, 利用可能になるまでのコスト ([100, 101] を参照)

(1.2.e) ストレージコスト: 2022 年 1 月における Bitcoin ブロックチェーンのデータサイズは, 約 390 GB [102] であり, 時間に対して線型的に増加するコスト ([101, 72] を参照)

(1.2.f) 電力消費: 合意形成を実行する際に必要な, 単位時間当たりのシステム全体の消費電力 ([103] を参照)

2. 社会課題

(2.a) ブロックチェーン管理者への継続的なインセンティブ確保: Bitcoin における, マイナーへの暗号通貨報酬を与える Coinbase トランザクション [21] の導入など

(2.b) 従来の商慣習や法制度との兼ね合い: Lightning Network [14] を用いた決済完了のタイミング, ブロックチェーンを用いたドキュメント存在証明 (Proof of Existence) [104] の有効性など

(2.c) ブロックチェーンでなくても実現できること: データベースの一貫性を保証する仕組みがあれば, 既存の権威組織の下でも, 同様なアプリケーションを開発可能か ([105] を参照)

ブロックチェーンだけでなく, どのようなシステムにおいても, 基本的なメカニズムだけではなく, 絶え間ない性能や信頼性向上の取組が不可欠である. しかし, ブロックチェーンは, 自律分散型の運営が本質である. また, 技術標準化に基づいて開発されているものではなく, 特定コミュニティーにおける実装が先行して行われるといった特性を持つ. 以上の 2 点から, ブロックチェーンの技術課題に対して精緻に取り組むことには, 本質的に困難である. 特に, 観測, 推定が困難で, 地理的に分散した合意形成に関与する, ブロックチェーンの P2P ネットワークのトポロジー構造の影響を強く受ける, 非機能要件についての研究 (1.2.a), (1.2.b) や (1.2.c) は, なかなか進展していない.

一方で, 上記のように分類されるブロックチェーン問題の中で, 性能やスケーラビリティは, ブロックチェーンの社会実装を阻害する大きな要因となっているため, その解決が強く望まれている. 本研究は, この (1.2.a), (1.2.b) や (1.2.c) の課題解決に対して, 数理的な解析手法をもって貢献する研究である. 第 4 章の分析においては, ブロック伝搬遅延がチェーン成長の振舞いに与える影響を, フォーク確率の観点から分析したことである. 自律分散型の運営で, ネットワークのトポロジー構造が自明でないブロックチェーンインフラストラクチャーにおいて, フォークと呼ばれる Bitcoin ブロックチェーン誕生以前にはなかった概念を用いた遅延分析を行ったことは, 大きな貢献と考えている.

また, 第 5 章においては, ブロックチェーンシステムの大きな課題であるスケーラビリティ問題を解決するために, 実質的なオンチェーントランザクションのスループットを改善する, 代表的なオフチェーン技術である ORU の分析を行った. ORU 利用者のオフチェーントランザクション待ち時間 $\mathrm{E}[W]$ は, 待ち行列モデルに基づいて評価される.

E[W] は, ORU システム全体のトランザクションスループットとのトレードオフがあり, その運用方法が土台となるブロックチェーンに与える影響もある. 集中管理され, 仕様通りに動作する従来の中央集権システムの分析と異なり, これらは, まさに, 現代のブロックチェーンシステムが抱える重要な課題である.

こうした非機能要件の課題解決のためには, PoW 以外の合意形成アルゴリズム (例えば, PoS (Proof of Stake) [33, 106], PBFT (Practical Byzantine Fault Tolerance) [32]) のチェーン成長の振舞い分析, 他のオフチェーン技術と類似した技術 (例えば, ブロックチェーンの分散データベースを複数のシャード (Shard) に分割し, 負荷を分散させるシャーディング (Sharding) [107]) の待ち行列理論に基づいた分析, 複数のブロックチェーンを利用して全体のスループットを改善する取組 (例えば, ブロックチェーンの相互接続性 [108] を高める研究や, 異なるブロックチェーン間の暗号通貨を交換可能にするアトミックスワップ [109]) の分析が必要不可欠である.

一方, 社会課題に対しては, まさに今取り組まれている様々な Web3 の世界観に基づいて, 仮想通貨だけでなく, NFT (非代替性トークン) や DAO (分散型自律組織), DID (分散型 ID), DeFi (分散型金融) など, 様々な取組がされている. Web3 が多くのケースで前提としているブロックチェーンは, 利便性向上のための日々のソフトウェアアップデートだけでなく, 合意形成の仕組みを大きく変更し, システム全体の性質を変える取組に対する議論が盛んに行われている. これらアプリケーション開発の実践の過程で, 社会課題も解決されていくことが期待される.

参考文献

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] Bitcoin GitHub. `https://github.com/bitcoin`.

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, vol. 151, 2014.

[4] A. Antonopoulus and G. Wood, *Mastering Ethereum: Building Smart Contracts and DApps*, O'Reilly Media Inc., 2019.

[5] Ethereum GitHub. `https://github.com/ethereum`.

[6] Hyperledger GitHub. `https://github.com/hyperledger`.

[7] Corda GitHub. `https://github.com/corda/corda`.

[8] Diem GitHub. `https://github.com/diem/`.

[9] MarketsandMarket. Blockchain Market with COVID-19 Impact Analysis, by Component (Platforms and Services), Provider (Application, Middleware, and Infrastructure), Type (Private, Public, and Hybrid), Organization Size, Application Area, and Region - Global Forecast to 2026. `https://www.marketsandmarkets.com/Market-Reports/blockchain-technology-market-90100890.html`.

[10] U. W. Chohan, "The Limits to Blockchain? Scaling vs. Decentralization," *Discussion Paper Series: Notes on the 21 st Century*, Feb. 2019.

[11] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. E. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song and R. Wattenhofer, "On scaling decen-

tralized blockchains (A position paper)," *International Conference on Financial Cryptography and Data Security*, 2016.

[12] R. Wattenhofer, *Blockchain Science: Distributed Ledger Technology 3rd Edition*, Independently published, 2019.

[13] Y. Kawase and S. Kasahara, "Transaction-Confirmation Time for Bitcoin: A Queueing Analytical Approach to Blockchain Mechanism," *Queueing Theory and Network Applications: QTNA 2017*, Lecture Notes in Computer Science, vol. 10591, Nov. 2017.

[14] J. Poon and T. Dryja, The bitcoin lightning network: Scalable off-chain instant payments, 2016. `https://lightning.network/lightning-network-paper.pdf`.

[15] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W/ Felten, "Arbitrum: Scalable, private smart contracts,"*Proceedings of the 27th USENIX Conference on Security Symposium*, pp. 1353-1370.

[16] Karl Floersch. Ethereum Smart Contracts in L2: Optimistic Rollup (2019). `https://medium.com/plasma-group/ethereum-smart-contracts-in-l2-optimistic-rollup-2c1cef2ec537`.

[17] Eth Hub. ZK-Rollups. `https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/zk-rollups/`.

[18] B. G. Gebraselase, B. E. Helvik and Y. Jiang, "Transaction Characteristics of Bitcoin," *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021.

[19] N. Johnson. ENS Documentation Release 0.1. `https://media.readthedocs.org/pdf/ens/latest/ens.pdf`.

[20] W. Vickrey, "Counter speculation, Auction, and Competitive Sealed Tenders,"

Journal of Finance, Vol. 16, No. 1, Mar. 1961.

[21] A. Antonopoulus, *Mastering Bitcoin*, O'Reilly Media Inc., 2017.

[22] Y. Sompolinsky, A. Zohar, "Secure high-rate transaction processing in bitcoin," *Proc. 19th Int. Conf. Financial Cryptogr. Data Secur. (FC ' 15)*, pp. 507-527, Jan. 2015.

[23] R. Pass, L. Seeman and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," *Advances in Cryptology - EUROCRYPT 2017*, Springer International Publishing, pp. 643-673, 2017.

[24] Quan-Lin Li, Jing-Yu Ma, Yan-Xia Chang, "Blockchain Queue Theory," *Springer Computational Data and Social Networks*, 2018.

[25] S. Eskandari and M. Moosavi, "SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain," *International Conference on Financial Cryptography and Data Security*, 2019.

[26] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach and A. Juels, "Flash Boys 2.0: Frontrunning, Transaction Reordering, and Consensus Instability in Decentralized Exchanges," *IEEE Symposium on Security and Privacy 2020*, 2020.

[27] H. Seike, Y. Aoki and N. Koshizuka, "Fork rate-based analysis of the longest chain growth time interval of a pow blockchain," *2019 IEEE International Conference on Blockchain*, 2019, pp. 253-260.

[28] H. Seike, Y. Aoki and N. Koshizuka, "Blockchain-based Scalable Ubiquitous Code Allocation Method Resilient to Congestion," *2021 IEEE International Conference on Blockchain*, 2021, pp. 272-279.

[29] H. Seike, T. Hamada, T. Sumitomo and Noboru Koshizuka, "Blockchain-based Ubiquitous Code Ownership Management System without Hierarchical Struc-

ture," *IEEE SmartWorld 2018*, Oct. 2018.

[30] H. Seike, Y. Aoki and N. Koshizuka, "Towards Smart Contracts for Verifying DNN Model Generation Process with the Blockchain," *2021 IEEE 6th International Conference on Big Data Analytics (ICBDA)*, 2021, pp. 160-168.

[31] C. Grunspan and R. Pérez-Marco, "Double spend races," *International Journal of Theoretical and Applied Finance*, 2018.

[32] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," *Proc. Usenix OSDI*, Berkeley, California. 1999.

[33] I. Bentov, A. Gabizon and A. Mizrahi, "Cryptocurrencies without proof of work," *International Conference on Financial Cryptography and Data Security*, pp. 142-157, 2016.

[34] V. Buterin, D. Reijsbergen, S. Leonardos and G. Piliouras, "Incentives in Ethereum's Hybrid Casper Protocol," *International Journal of Network Management*, 2020.

[35] I. Bashir, *Mastering Blockchain - Second Edition*, O'Reilly Media Inc., 2019.

[36] V. Buterin. Merkling in Ethereum. `https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/`.

[37] M. Al-Bassam, A. Sonnino and V. Buterin, "Fraud Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities," *Scaling Bitcoin 2018*, Oct. 2018.

[38] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Conference on the Theory and Application of Cryptographic Techniques - CRYPTO '87*, pp. 369-378, Springer-Verlag, 1987.

[39] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains," *Cryptoeconomics and Security Conference 2017*, 2017.

[40] C. Decker and R. Wattenhofer, "Information propagation in the Bitcoin network," *IEEE P2P 2013 Proceedings*, Trento, 2013, pp. 1-10.

[41] I. Eyal, E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable", *Financial Cryptography*, 2014.

[42] S. Bag, S. Ruj and K. Sakurai, "Bitcoin Block Withholding Attack: Analysis and Mitigation," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1967-1978, Aug. 2017.

[43] M. Rosenfeld, "Analysis of hashrate-based double spending," *ArXiv 1402.2009v1*, Feb. 2014.

[44] J. A. Garay, A. Kiayias, N. Leonardos, "The bitcoin backbone protocol: Analysis and applications", *Proc. 34th Int. Conf. Theory Appl. Cryptogr. Techn. (EUROCRYPT' 15)*, pp. 281-310, Apr. 2015.

[45] S. Zhang and J. -H. Lee, "Double-Spending With a Sybil Attack in the Bitcoin Decentralized Network," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 10, pp. 5715-5722, Oct. 2019.

[46] M. Iqbal and R. Matulevičius, "Exploring Sybil and Double-Spending Risks in Blockchain Systems," *IEEE Access*, vol. 9, pp. 76153-76177, 2021.

[47] K. Pearson, "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *Philosophical Magazine Series 5,* 50(302): 157- 175, 1900.

[48] S. Kullback and R. A. Leibler, "On information and sufficiency," The Annals of Mathematical *Statistics*, vol. 22, no. 1, pp. 79-86, 1951.

[49] M. Sugiyama, T. Suzuki and T. Kanamori, *Density Ratio Estimation in Machine Learning*, Cambridge University Press, 2012.

[50] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering bitcoin's public topology and influential nodes," 2015.

[51] R. Bowden, H. P. Keeler, A. E. Krzesinski and P. G. Taylor, "Block arrivals in the bitcoin blockchain", arXiv, 2018.

[52] H. Heo and S. Shin, "Understanding Block and Transaction Logs of Permissionless Blockchain Networks", *Security and Communication Networks*, vol. 2021, 2021.

[53] S. Šimunić, D. Bernaca and K. Lenac, "Verifiable Computing Applications in Blockchain," *IEEE Access*, vol. 9, pp. 156729-156745, 2021.

[54] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, NewYork: Dover, 1972.

[55] J. D. C. Little, "A proof of the queueing formula $L = \lambda W$," in *Operations Research*, vol. 9, pp. 383-387, 1961.

[56] W. Feller, *An Introduction to Probability Theory and Its Applications*, NewYork: Wiley, 1971.

[57] N. Koshizuka and K. Sakamura, "Ubiquitous ID: Standards for Ubiquitous Computing and the Internet of Things," *IEEE Pervasive Computing*, vol. 9, no. 4, pp. 98-101, Oct.-Dec. 2010.

[58] ITU-T, *Multimedia information access triggered by tag-based identification*, International Telecommunication Union Recommendation H. 642, Jun. 2012.

[59] C. Ishikawa, "A URN namespace for ucode", *RFC 6588*, Apr. 2012.

[60] ICANN Blog, FAQs for RegisterFly customers. `https://www.icann.org/news/blog/faqs-for-registerfly-customers`.

[61] Namecoin. `https://namecoin.org/`.

[62] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A Global Naming and Storage System Secured by Blockchains," *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 181–194, Denver, CO, June 2016. USENIX Association.

[63] ENS. `https://ens.domains/`.

[64] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292-2303, 2016.

[65] H. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of Namecoin and lessons for decentralized namespace design," *WEIS ' 15: Proceedings of the 14th Workshop on the Economics of Information Security*, June. 2015.

[66] Blockstack GitHub: issue 1052, *Add more warnings about keeping computer on during name registration*, `https://github.com/blockstack/blockstack-browser/issues/1052`.

[67] EIP 162, *Initial ENS Registrar Specification*, `https://github.com/ethereum/EIPs/issues/162`.

[68] M. Inoue How to Get Back an Old .ETH Name Deposit. `https://medium.com/the-ethereum-name-service/how-to-get-back-an-old-deposit-1e2b1767b930`, Aug. 2020.

[69] TRON Forum, Ubiquitous ID Center, *ucode: Ubiquitous Code*, 910-S101/UID-00010, Jul. 2009.

[70] MySQL Documentation, *MySQL 5.7 Reference Manual*, `https://dev.mysql.com/doc/refman/5.7/en/example-auto-increment.html`.

[71] EIP 1559, *Fee market change for ETH 1.0 chain*, `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md`.

[72] R. Nagayama, R. Banno and K. Shudo, "Trail: A Blockchain Architecture for Light Nodes," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1-7.

[73] EIP 155, *Simple replay attack protection*, `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-155.md`.

[74] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," *NIPS*, 2014.

[75] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[76] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[77] Q. Wang, W. Guo, K. Zhang, A. G. O. II, X. Xing, X. Liu, and C. L. Giles, "Adversary resistant deep neural networks with an application to malware detection," *Proc. of KDD*, 2017.

[78] T. Gu, K. Liu, B. Dolan-Gavitt and S. Garg, "BadNets: Evaluating Backdooring Attacks on Deep Neural Networks," *IEEE Access*, vol. 7, pp. 47230-47244, 2019.

[79] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, " Neural cleanse: Identifying and mitigating backdoor attacks in neural networks, " *40th IEEE Symposium on Security and Privacy*, 2019.

[80] K. Crawford, R. Dobbe, T. Dryer, G. Fried, B. Green, E. Kaziunas, A. Kak, V. Mathur, E. McElroy, A. N. Sánchez, D. Raji, J. L. Rankin, R. Richardson, J. Schultz, S. M. West and M. Whittaker, AI Now 2019 Report. New York: AI Now

Institute, 2019, https://ainowinstitute.org/AI_Now_2019_Report.html.

[81] Justice in Forensic Algorithms Act of 2019, H. R. 4368, https://www.congress.gov/bill/116th-congress/house-bill/4368/text?r=2&s=1.

[82] PyTorch master documentation v1.1.0, *Reproducibility*, https://pytorch.org/docs/1.1.0/notes/randomness.html.

[83] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by back propagating errors," *Nature*, Vol. 323, pp. 533-536, 1986.

[84] L. Shi, H. Luo, X. Yang and Y. Sun, "A High-availability Data Backup Strategy for IPFS," in *2019 IEEE International Conference on Consumer Electronics*, Taiwan, 2019.

[85] S. Gueron, S. Johnson and J. Walker, "SHA-512/256," *2011 Eighth International Conference on Information Technology: New Generations*, Las Vegas, NV, 2011, pp. 354-358.

[86] PyTorch master documentation v1.1.0, *CUDA Semantics*, https://pytorch.org/docs/1.1.0/notes/cuda.html.

[87] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *MICCAI 2015*, pp. 234-241, May. 2015.

[88] Y. CuLun, C. Cortes and C. J.C. Burges, "THE MNIST DATABASE of handwritten digits," http://yann.lecun.com/exdb/mnist/.

[89] P. X. Huang, B. B. Boom and R. B. Fisher, "Fish recognition ground-truth data," http://groups.inf.ed.ac.uk/f4k/GROUNDTRUTH/RECOG/.

[90] R. Canetti, B. Riva and G. N. Rothblum, "Practical Delegation of Computation Using Multiple Servers," *Proc. 18th ACM Conf. CCS*, pp. 445-454, 2011.

[91] F. Tramèr and D. Boneh, "Slalom: Fast verifiable and private execution of neural networks in trusted hardwarem," in *ICLR 2019*, 2019.

[92] Z. Ghodsi T. Gu and S. Garg "SafetyNets: Verifiable execution of deep neural networks on an untrusted cloud," *in Advances in Neural Information Processing Systems*, pp. 4675-4684, 2017.

[93] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang and W. Luo, "DeepChain: Auditable and Privacy-Preserving Deep Learning with Blockchain-based Incentive," in *IEEE Transactions on Dependable and Secure Computing*, 2019.

[94] R. Canetti, B. Riva and G. N. Rothblum, "Refereed delegation of computation," *Information and Computation*, vol. 226, pp. 16-36, 2013.

[95] B. Parno and C. G. M. Raykova, "Pinocchio: Nearly Practical Verifiable Computation," `https://eprint.iacr.org/2013/279.pdf`.

[96] C. Lund, L. Fortnow, H. Karloff and N. Nisan, "Algebraic methods for interactive proof systems," *in Journal of the ACM*, pp. 859–868, 1992.

[97] P. Micikevicius, S. Narang, J. Alben, G. F. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh and H. Wu, "Mixed precision training," in *ICLR 2018*, 2018.

[98] E. G. Weyl, P. Ohlhaver and V. Buterin, "Decentralized society: Finding web3's soul", 2022.

[99] M. Bartholic, A. Laszka, G. Yamamoto and E. W. Burger, "A Taxonomy of Blockchain Oracles: The Truth Depends on the Question," *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2022, pp. 1-15.

[100] D. Khan and L. T. Jung and M. A. Hashmani, "Systematic Literature Review of Challenges in Blockchain Scalability," *Applied Sciences*, 2021.

[101] D. S. Gadiraju, V. Lalitha and V. Aggarwal, "Secure Regenerating Codes for Reducing Storage and Bootstrap Costs in Sharded Blockchains," *IEEE International Conference on Blockchain*, 2020.

[102] Blockchain.com. Explorer: Blockchain Size (MB). https://www.blockchain.com/charts/blocks-size.

[103] V. C. Coroamă, "Exploring the Energy Consumption of Blockchains through an Economic Threshold Approach," *2021 Joint Conference - 11th International Conference on Energy Efficiency in Domestic Appliances and Lighting & 17th International Symposium on the Science and Technology of Lighting (EEDAL/LS:17)*, 2022, pp. 1-10.

[104] S. Chiliveri, J. Grandhi, M. Uttam Patil, L. E. P.R. and M. Ethirajan, "ProveDoc: A Blockchain Based Proof of Existence with Proof of Storage," *2019 International Conference on Information Technology (ICIT)*, 2019, pp. 239-244.

[105] K. Wüst and A. Gervais, "Do you Need a Blockchain?," *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2018, pp. 45-54.

[106] E. Deirmentzoglou, G. Papakyriakopoulos and C. Patsakis, "A Survey on Long-Range Attacks for Proof of Stake Protocols," *IEEE Access*, vol. 7, pp. 28712-28725, 2019.

[107] S. Kantesariya and D. Goswami, "Determining Optimal Shard Size in a Hierarchical Blockchain Architecture," *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020.

[108] Monika and R. Bhatia, "Interoperability Solutions for Blockchain," *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, pp. 381-385, 2020.

[109] M. Herlihy, "Atomic cross-chain swaps," *Proc. ACM Symp. Principles Distrib. Comput. (PODC)*, pp. 245-254, 2018.

付 録A

A.1 PoW ブロックチェーンのマイニング成功時間の分布

本節では, C. Grunspan ら [31] が前提とした「マイニング成功時間が指数分布に従う」ことの証明をレビューとして行う. 時刻 0 にマイニングを開始し, マイニングに初めて成功するまでにかかる時間を t とする. この確率密度関数 (PDF) を $g(t)\ \ t \geq 0$ とする. $g(t)$ を用いることで, 時刻 t から少し経過した時間, 例えば Δt の間に, マイニングが初めて成功する確率は $g(t) \cdot \Delta t$ と近似することができる. 従って, $g(t)$ が連続で, 時刻 $t > 0$ の至る所で積分可能であるならば, 任意時刻 t_1 から t_2 $(0 < t_1 < t_2)$ の間にマイニングが成功する確率 $P(t_1, t_2)$ は, 次式で表される.

$$P(t_1, t_2) = \int_{t_1}^{t_2} g(t)\ dt. \tag{A.1}$$

$t_1 = 0, t_2 = t$ とした場合, $P(t_1, t_2)$ は時刻 0 から t までの間にマイニングが成功する確率となる. $P(t_1, t_2)$ を $P(0, t) = G(t)$ として累積分布関数 $G(t)$ と定義する.

累積分布関数を用いて, マイニングの成功確率 (ある種の数学的問題を解く確率) が, 時間に依存せず一定であるという仮定を置けば, 次の微分方程式が成立する.

$$\begin{cases} G(t + \Delta t) - G(t) & = (1 - G(t)) \cdot G(\Delta t), \\ G(\Delta t) & = G(0) + G'(0) \cdot \Delta t + O((\Delta t)^2). \end{cases} \tag{A.2}$$

第一式は, 時刻 t から $t + \Delta t$ の間に初めてマイニングが成功する確率である. t から $t + \Delta t$ に初めてマイニングが成功するという事は, 言い換えると時刻 t までマイニングが成功していないという事である. そして, マイニングが成功する確率は時間に依存しないので, 時刻 0 から Δt の間にマイニングが成功する確率と, 時刻 t までにマイニン

グが成功していないという前提を置いた場合の時刻 t から $t+\Delta t$ にマイニングが成功する確率は等しくなる．第二式は, 十分小さい Δt を置けば, 確率密度関数の定義から $G(\Delta t) \simeq \int_0^{\Delta t} g(t)\ dt \simeq g(0) \cdot \Delta t$ となる事を表している．

第二式を第一式に代入して, 微分方程式を解く．この際, $G'(0) = \alpha$ としておけば, $g(t) = G'(t) = \alpha \exp(-\alpha t)$ なる式が得られる．α が大きいほど $g(t)$ は $t \neq 0$ で高速に 0 に収束していく．つまり, α はマイニングの実行速度 (マイニングレート) を決定するパラメータといえる．また, マイニングに成功する平均時間 $\bar{t}$ は $\bar{t} = \int_0^{\infty} t \cdot g(t)\ dt$ で計算され, $\bar{t} = 1/\alpha$ となる．これが, Bitcoin におけるマイニング実行の平均時間 (600 秒 = 10 分) である．Bitcoin では, $\alpha = 1/600$ となるようにマイニングの難易度を調整している．

A.2 PoW ブロックチェーンのフォーク確率の導出

本節では, ブロック生成速度が高い場合に用いることができるフォーク確率を導出する．C. Decker らの論文 [40] において, ブロックチェーンのフォークする確率 P_{Fork} は次式で定義されている．この式はネットワークのブロック生成速度 (マイニングレート) α が十分に小さいことを前提としている．

$$P_{\mathrm{Fork}} = 1 - (1 - P_b)^{\int_0^{\infty}(1-F(t))dt}. \tag{A.3}$$

ここで P_b は, 単位時間である 1 秒の間にブロックが生成される確率である．あるブロック高 h で最初にマイニングされたブロック $b_0(h)$ とする．$F(t)$ は, $b_0(h)$ がマイニングされた時刻を 0 として, フォークが発生しないという条件の下で, $b_0(h)$ をマイニングしている計算リソースの割合である．

このように各種変数を定義すると, P_b の確率は以下のように計算できる．

$$P_b = \int_0^1 \alpha \cdot \exp(-\alpha t)dt = 1 - \exp(-\alpha). \tag{A.4}$$

ここで, α はネットワーク全体のブロック生成速度を表すパラメータである．また, C. Decker らは, Bitcoin のコンセンサス時間が平均 600 秒程度と長いこと (つまり,

$\alpha \approx 1/600$) を利用して, さらに式 A.6 を次のように近似している.

$$P_b = 1 - \exp(-\alpha) \approx 1 - (1 - \alpha) = \alpha. \tag{A.5}$$

しかし, Ethereum などのブロック生成時間の平均は 14 秒程度であり, Bitcoin のそれと比較して短い. 従って, 本節の導出では A.4 で計算されるフォーク確率を採用する.

次に, マイニングされる時刻を十分に小さな Δt (> 0) 単位で区切った場合に, どのくらいの確率でマイニングが成功するかについて考える. 分割された各時刻を t_i ($t_i \geq 0$, $i \in \mathbf{N}$) と定義し, 時刻 $[t_i, t_{i+1}]$ の間にマイニングされる確率 $P_b([t_i, t_{i+1}])$ を考えると次式のように近似できる (X_b はブロックがマイニングされる時刻).

$$\begin{aligned} P_b([t_i, t_{i+1}]) &= \Pr(X_b < t_{i+1} \mid X_b \geq t_i), \\ &\approx 1 - \exp(-\alpha(1 - F(t_i)) \cdot \Delta t). \end{aligned} \tag{A.6}$$

次に, ブロックのマイニングを時刻 0 に開始して, フォークが発生しない確率について考える. 仮にフォークが発生しない場合, 式 A.6 が表す事象が任意の区間 i において成立しないということである. フォークが発生する確率 Pr[Fork] は, その余事象であるフォークが発生しない確率を用いて計算可能なため次式から得られる.

$$\begin{aligned} \Pr[\text{Fork}] &\approx 1 - \prod_{i=0}^{\infty}(1 - (1 - \exp(-\alpha(1 - F(t_i)) \cdot \Delta t)), \\ &= 1 - \prod_{i=0}^{\infty} \exp(-\alpha(1 - F(t_i)) \cdot \Delta t), \\ &= 1 - \exp(-\alpha \sum_{i=0}^{\infty}(1 - F(t_i)) \cdot \Delta t), \\ &= 1 - \exp(-\alpha \int_0^{\infty}(1 - F(t))dt), \quad (\Delta t \to 0). \end{aligned} \tag{A.7}$$

この PoW ブロックチェーンのフォーク確率は, あるブロック高において 2 つ以上のブロックが管理者ノード (マイナー) によって生成されることにより, 自然に発生するケースのみを考慮している. 利己的なマイニング (Selfish Mining) [41, 42] による人為的にフォークが発生するケースは, 考慮していないことに注意する.

A.3　式 4.10 の導出

$$\int_0^\infty g_{\mathrm{nd}}(t)\left(\frac{g_{\mathrm{lb}}(t)}{g_{\mathrm{nd}}(t)}-1\right)^2 dt = \int_0^\infty \alpha\cdot\exp(-\alpha t)dt$$
$$+\int_{T_w}^\infty\Big(-2\alpha C\exp(-\alpha t+\alpha T_w\log t)\big(1-\frac{T_w}{t}\big)$$
$$+\alpha C^2\exp(-\alpha t+2\alpha T_w\log t)\big(1-\frac{T_w}{t}\big)^2\Big)dt$$
$$\big(T_w=\frac{-\log(1-P_F)}{\alpha}\ ,\ C=\exp\big(\alpha\cdot T_w(1-\log\ T_w)\big)\big) \tag{A.8}$$

式 A.8 は下記の 3 式を用いることで, 式 4.10 のように簡略化することができる.

$$\int_{T_w}^\infty \exp(-at+b\log t)\cdot(1-\frac{c}{t})^2dt = a^{-(b+1)}\big((ac)^2\Gamma(b-1,aT_w)$$
$$-2ac\Gamma(b,aT_w)+\Gamma(b+1,aT_w)\big) \tag{A.9}$$
$$\int_{T_w}^\infty \exp(-at+b\log t)\cdot(1-\frac{c}{t})dt = -a^{-(b+1)}\left(ac\Gamma(b,aT_w)-\Gamma(b+1,aT_w)\right) \tag{A.10}$$
$$\Gamma(a+1,x)=a\Gamma(a,x)+x^a\exp(-x) \tag{A.11}$$

A.4　式 4.11 の導出

$$\int_0^\infty t\cdot g_{\mathrm{lb}}(t)dt = \int_0^{T_w} t\cdot 0\ dt+\int_{T_w}^\infty t\cdot\alpha C\exp(-\alpha t+\alpha T_w\log t)\big(1-\frac{T_w}{t}\big)dt$$
$$=\int_{T_w}^\infty \alpha C\exp(-\alpha t+(\alpha T_w+1)\log t)\big(1-\frac{T_w}{t}\big)dt \tag{A.12}$$
$$\big(T_w=\frac{-\log(1-P_F)}{\alpha}\ ,\ C=\exp\big(\alpha\cdot T_w(1-\log\ T_w)\big)\big)$$

上式は, 式 A.10 を用いることで, 式 4.11 のように簡略化することができる.