Highway：フレキシブルなファイナリティによる効率性の高いコンセンサス

Daniel Kane[1]、Andreas Fackler[2]、Adam Gągol[3]、Damian Straszak[4]
[1]カリフォルニア大学サンディエゴ校コンピューター理工学部
[2]CasperLabs 株式会社
[3,4]Cardinal Cryptography

2021 年 1 月 16 日

要旨

効率性が高く、部分的かつ同期的な BFT（Byzantine Fault Tolerance、ビザンチン故障耐性）を有するコンセンサス・プロトコルの設計において、昨今多くの進歩が見られているが、このプロトコルが目的としているのは、プルーフ・オブ・ステークのブロックチェーン・システム用のコアなコンセンサス・エンジンとして機能することである。最先端の各種ソリューションが、この理論モデルの下での実質的な最適性能を達成していく一方で、こうしたシステムの実用面のいくつかは、このモデルでの攻略ができておらず、依然として改善の余地があると言える。中でも最も注目すべきなのは、通常の実行時において、圧倒的な割合のノードが、こうしたシステムの金銭的なインセンティブにより善意的にプロトコル・ルールに従う反面、そのうちのごく少数のノードは、おそらくは一時的なネットワーク上の問題により不正を行う可能性があると考えられる点である。直感的な話をすると、ほぼ全てのノードが善意的に動作するという事実から考えれば、こういった期間にファイナライズされるブロックに対して強い信頼性を持てるはずであるが、ファイナリティがバイナリーである従来型のモデルの下ではこの考え方は通用しない。

本論文では、部分的かつ同期的な従来の BFT モデルにおいて safety（安全性）と liveness（生存性）を維持する新たなコンセンサス・プロトコルである Highway を提示し、同時に、既存ソリューションに対する実用面での改良を加えていく。具体的に説明すると、Highway におけるブロックのファイナリティはバイナリーではなく、ブロックをリバートする為にプロトコル・ルールを破る必要があるノードの割合で示される。善意からの参加が行われている期間中に、ブロックのファイナリティは 3 分の 1 をはるかに超え（従来型のプロトコルの最大値として）、1（すなわち完全なる確実性があることを示す）にさえ到達する場合もある。ファイナリティをこのように定義することで、Highway はプロトコルを実行するノード間のセキュリティ閾値の構成に関する flexibility（柔軟性）を提供し、これにより、高いレベルの信頼性を必要とするノードよりも、低い閾値を有するノードがより速くファイナリティ―に到達できるようになっている。

1 序論

Bitcoin[Nak08]や、分散型で改ざん耐性を有するデータベース――すなわちブロックチェーン――の概念が導入されて以来、これらデータベースを設計する目的で、多種多様なパラダイムが開発されてきた。

最近では、こうした PoS（プルーフ・オブ・ステーク）に基づいたシステムを構築するという着想が広く関心を集めている。参加促進やシステム確保に使用する元来の PoW（Bitcoin で用いられているプルーフ・オブ・ワーク）のメカニズムにおいては、参加者の投票権（voting power）は、所有する計算能力の量に比例するが、その一方で PoS においては、投票権はトークン（本システムに固有のデジタル通貨）の量に比例する。定期的に参加者による固定サイズの committee を委任し、この committee がブロックチェーンに追加するブロックに関するコンセンサス実施の任を負うといったあり方が、こういったシステムにおいて広く支持され選択されている。このブロックチェーンの構築方法は、以下 2 つの点で Bitcoin などの vanilla PoW よりも本質的に優れていると言える。：1）過去 40 年間に開発された従来の許可型（permissioned）コンセンサス・プロトコルの 1 つを実行することができる。2） ノードに対して参加についてのリワードを与えるだけでなく、違反した committee メンバーに対しては保証金を削減して不正行為へのペナルティを与えることができる。

近頃は、こうした PoS ブロックチェーンでコアエンジンとして利用可能な許可型のコンセンサス・プロトコルの設計が飛躍的に進歩している。[AMN+19、BKM18、BG17、CS20、GLSS19、GAG+19、YMR+19、ZRAP18] その圧倒的多数が、部分的かつ同期的な BFT モデル[DLS88]において設計されており、本モデルではノード間の通信が最終的に同期となり、所与のノードの割合――例えば（本モデルでの最適値であれば）3 分の 1――未満が不正となり、任意にプロトコルに違反する可能性もあるとされている。Hotstuff [YMR+19]、Tendermint [BG17]、および Streamlet [CS20]等の最先端のプロトコルは、回線容量、ファイナライズのレイテンシー（待ち時間）、さらには重要なポイントとして、簡潔性の面でも、最適と言えるレベルにまで達している。とはいえ、こうしたブロックチェーン・システムにおいては、従来型のモデルでは攻略しきれていない実用面での特性がいくつか存在し、結果として改善の余地がかなり残っている。その重要な側面の 1 つとして挙げられるのが、ノードを、善意的なものとビザンチン故障性のあるものに分けたとしても、それが真の状況を反映していない場合もあるという点である。実際、本モデルに従った場合、DDoS（Denial-of-service,サービス拒否）攻撃や一時的なネットワーク障害のせいでプロトコル・メッセージをいくつか見逃してしまったような「善意の」ノードさえも、ビザンチン故障性があると判断されてしまう。3 分の 1 を超えるノードが、そうしたネットワークの問題で（たった数秒間でも）障害の危険にさらされている状況においては、従来型の BFT モデルのプロトコルが正常に機能する保証はない。

一方で、時折発生するこうしたオフラインでの一定の期間は別にして、実世界システムにおいては、全てではないにしても圧倒的な割合のノードが善意的にプロトコル・ルールに従っていると想定するのが妥当である。これは、善意の参加への金銭的インセンティブがもたらす結果だ。確かに、ノードは善意の仕事をすることで給与を支給されているのであり、オフラインでいたり、プロトコルの進歩への貢献が不十分であったりすればペナルティを科せられる為、committee のメンバーにとっては、ノードがコンセンサス・プロトコルに対して積極的な参加を行っているかを確認することは大きなメリットにつながる。実際に、プロトコル違反へのペナルティが存在する為、敵対者が、成功する保証もなく重大な損失を負うリスクにもつながる攻撃を試みる可能性は非常に低い。したがって、システム全体をダウンさせることを企図した大規模な組織的攻撃を唯一の例外として、ほぼ全てのノードは善意的に動作すると常に心に留めておくべきであると言える。

こうした状況の実現が刺激となり、従来の意味での safety（安全性）を維持するプロトコルの設計の研究が数件程行われてきており、同時に、「典型的な」シナリオにおいてより良い保証を提供する試みも続けられている。本論文では、この分野の研究に資する新たなプロトコル――Highway――を提示していく。Highway のセキュリティは、依然として部分的かつ同期的な BFT モデルに基づいて形式化している為、具体的に言ってみれば、全てのノードの 3 分の 1 にビザンチン故障性があるというような、最も厳しい設定であっても safety（安全性）と liveness（生存性）を達成できる。また一方で、それに加え、Highway は以下の二機能を提供しており、これらの機能ゆえに、実世界での展開という面で特に注目を呼んでいる。第一の機能としては、この Highway により、ノードの大部分が善意的な参加を行う期間において、典型的な閾値である 3 分の 1 よりはるかに高い「confidence（信頼性）」で、ブロックがファイナリティに到達可能となる。一例を挙げてみると、ブロックが 0.8 のファイナリティ confidence（Highway で可能な値）に到達する場合、ノードの 80%以上が、同ブロックをチェーンからリバートするのにプロトコルに違反する必要がある。これは、ファイナライゼーションの従来の概念、つまり、ブロックがファイナライズされるのか（つまり 3 分の 1 のファイナリティ confidence）、

それともファイナライズされないのかというバイナリーの概念とは対照的である。Highway における実用面での第二の改良点は、[MNR19]において定義されている概念に似た flexibility（柔軟性）を実現している点である。Highway に参加しているノードは、プロトコルにおいて、許容数のビザンチン故障性のあるノードとクラッシュ障害性のあるノード（オフラインになる可能性があるが、それ以外は善意的なノード）との間で、異なるセキュリティのトレードオフで構成される可能性がある。したがって、flexibility が意味することというのは、これらの構成面での違いがあるにもかかわらず、全てのノードが単一のバージョンのプロトコルを実行して同じアクションを実行し、ノードが行うファイナリティの決定のみが、選択されたパラメーターに依存しているという状態を意味する。実際的な結果として見ると、セキュリティ閾値が低いノードは、閾値が高いノードよりもはるかに速くファイナリティに到達する可能性があるが、これらどちらのノードもその想定を満たしている限りにおいて、同一のブロックをファイナライズし、一致し続ける。

技術的には、Highway は、DAG（Directed acyclic graph、有向非巡回グラフ）ベースのプロトコル[Bai16、GLSS19、MMS99、ZRAP18]として分類することができ、このプロトコルにおいて、ノードは、因果関係の順序を示す有向非巡回グラフを形成しながら、プロトコル・メッセージの共通の履歴を合同で維持する。Highway はその設計上、CBC-Casper アプローチ[ZRAP18]に由来しており、そこに新しいファイナリティ・メカニズム、メッセージ作成スケジュール、およびスパム防止メカニズムを使用することにより大幅な改良が加えられている。Highway プロトコルには、実用面で望まれる機能と共に、DAG ベースのプロトコルという概念面での簡潔性もある為、プルーフ・オブ・ステークをベースとしたブロックチェーンでコンセンサス・エンジンを求めるならば、Highway は堅実な選択肢の 1 つと考えられる。

2 結論

2.1 モデル

本論文においては、n個のバリデーター（検証者）の固定された集合$\mathcal{V}$を有するシステムが一式あると考えることとし、その各々のバリデーターが、他のノードに既知の公開鍵を装備しているとする。このモデルは「許可型ブロックチェーン（permissioned blockchain）」のシナリオと一致（match）するものの、本プロトコルは、本バリデーターの集合を循環（rotate）させることにより、半自由参加型（semi-permissionless）のシナリオにも適応させることができる。本論文のモデルにおいては、以下のような仮定を設定している。：

- （*信頼できるポイント対ポイントの通信*）本論文においては、チャネルがメッセージをドロップ（drop）することはなく、本プロトコル内の全てのメッセージは送信者のデジタル署名によって認証されると仮定する。

- （*部分的かつ同期的なネットワーク*）公知の限界時間Δと、未知のGlobal Stabilization Time（GST）が存在し、これにより、GST以後は、バリデーターがメッセージを送信するたびに、そのメッセージがΔの時間内に、受信者に届く。さらに本論文では、バリデーターには限界が設けられたクリック変動[1]があると仮定している。こうした部分的かつ同期的なバージョンは、*既知のΔflavour*として知られているが、公知のΔのないバージョンに関する議論については、小節3.6.3を参照のこと。

[1] 限界が設けられたクロック変動は、[DLS88]などのビザンチン故障性のクロック同期化を利用して、任意の部分的かつ同期的なネットワークにおいて達成可能である点に留意のこと。

- （*ビザンチン故障*）本論文では、n個のバリデーターの一部であるfが、任意の敵対者の完全な制御下にあり、したがって、fが本プロトコルから任意に逸脱することができると仮定する。safety（安全性）とliveness（生存性）は、別々の限界を必要とし、後者はクラッシュ障害性のあるノードの数との相互作用も持つ為、fとnの関係については、本論文ではグローバルな仮定は行わない。

- （*クラッシュ障害*）本論文では、n個のノードの一部であるcが、本プロトコル実行のある時点において、永久に無反応となる可能性があると仮定している。

2.2 ブロックチェーンのコンテクストにおけるコンセンサス

一般的なブロックチェーン・システムにおいては、外部環境から受け取るトランザクションで形成され、絶え間なく成長し続けるチェーンに対して、バリデーターには反復コンセンサスを実行する任務が課せられている。その過程において、バリデーターは、トランザクションを、ブロックチェーンを形成するブロックの中に囲い込み、そのブロックチェーンの中で各ブロックは、ハッシュによってその前身を参照する。最初のブロックである*ジェネシス・ブロック*は、プロトコル定義の一部である。

Highway では、バリデーターの集合は一定であり、非常に制御された変更（異なる era の間の変更。小節 4.2 を参照のこと。）のみの影響を受ける状態である為、所与のタイムスロットでのブロックの構築については、特定のバリデーターが直接任命を受ける。ローカルのキューからのトランザクションと、その前身であるはずのブロックのハッシュを囲い込むことによって、バリデーターはブロックの構築を行う。

バリデーターが最後に構築されたブロックを（意図的に、あるいはネットワーク障害が原因で）参照しない場合もある為、ブロックの集合$\mathcal{B}$は、各ブロックからルートへ向かう固有のパスである、ジェネシス・ブロックGを伴う１つのツリーとなっている。こうしたシナリオにおけるコンセンサス・プロトコルの主要な目的は、こういったツリーの中から単一の分岐を選択することだ。ブロックBの場合、本論文では、他の分岐上のブロックのどれかが選択されるとブロックBを選択することはできない為、ブロックBと*競合する*ものとして、他の分岐上の全てのブロックを参照する。

2.3 実用面での課題

楽観的アプローチにも応える強力なファイナリティ――Dwork、Lynch、およびStockmeyer [DLS88]による部分的かつ同期的なモデルの初期定義がなされて以来、同設定におけるプロトコルの多様なパラメーターの最適化を目的として、膨大な量の研究が創出されてきた。しかしその一方で、そのほとんどの研究は、$n/3$個を超える不正なノードの存在が、そうしたプロトコルに対する証明可能な保証の存在よりも前にある、という準公式な仮定の下で書かれている。この仮定は、元の論文でも証明した通り、$n < 3f + 1$である場合に、部分的かつ同期的なプロトコルが liveness（生存性）とファイナリティの双方を保証することは不可能であるという事実に由来している。

しかしながら、こうした否定的な結果にもかかわらず、不正なノードでも長期間にわたって同プロトコルから逸脱せずブロックのファイナライズに積極的に取り組むといったシナリオにおいて、追加的なファイナリティ保証を提供することは可能である。従来型のセキュリティ・モデルの観点からすると、注目に値する重要性などないと捉えられてしまう可能性もあるが、本論文では、以下は実用面での重大な結果をもたらす点であると

強調する。――すなわち、ブロックチェーンを展開する際のほとんどのケースでは、大抵の場合において、全てのノードは基本的に、積極的に協力しあってコンセンサスを達成すると仮定することができる。[2]したがって、こうした期間に形成されたブロックに対して非常に強力なファイナリティ保証を提供することが可能なのであり、それゆえに、それをリバートするには、$n/3$をはるかに超えるバリデーターが結託しなくてはならないということになるのである。

Flexibility（柔軟性）――以前示した$n < 3f + 1$という限界を一旦除外して考えた場合、ファイナリティと liveness の間にはごく自然なトレードオフが発生する。すなわち、より強力なファイナリティ保証を求めれば求めるほど、より多くのバリデーターが善意で協力しあい、同保証を用いてブロックをファイナライズしていく必要があるということだ。このトレードオフは、全てのバリデーターが、これから使用する共通のファイナリティ閾値に対して合意することで解消できる場合もあるが、その解決方法には、重大な制限が存在する可能性が高い。

しかしながら Highway では、共通の閾値で合意する必要がない為、全てのバリデーターが別個の閾値、あるいは複数の異なる閾値を使用することも可能なのである。この特定のハイパーパラメーターに関する追加的なコンセンサスを形成する必要がないという事実に加え、こうした機能が内包する重要な意味の 1 つとして挙げられるのは、Highway においては、バリデーターはそのエコシステムにおいて、各々が若干異なる役割を果たすことが可能であるという点である。――例えば、比較的小規模なトランザクションを主に扱っていて、その各ケースで、非常に高レベルのセキュリティよりは短いレイテンシー（待ち時間）を重要視する状況に置かれたバリデーターが存在する場合もあれば、（通常、より高い閾値に到達するには時間がかかる為、同プロトコルが提示された後ではこの状況は顕著になる）レイテンシーよりも safety（安全性）を優先するバリデーターが存在する場合もあるという訳である。[3]

2.4 CasperLabs による貢献

本論文で提示する Highway は、楽観的アプローチに応える強力なファイナリティを達成するコンセンサス・プロトコルであり、バリデーターが各々異なる confidence 閾値を使用して所与のブロックを必ず「ファイナライズ」できると確信がもてるようにすることにより、flexible な（柔軟性がある）ものとなっている。（confidence 閾値とファイナリティについては、双方とも小節 3.3 において適切に定義する。）

一部のバリデーターが積極的に同プロトコルから逸脱しない限りは、ブロックのファイナリティは、所与のバリデーターに対してのみ増加していく可能性もあり、このバリデーターは、PoW のシナリオにおいて絶え間なく増え続けるブロックの確認に対して直感的に対応する。ただし、PoW とは異なり、Highway では、所与のブロックに対する confidence の水準は、こうしたブロックを元に戻す目的で不適切な動作をしなくてはならないバリデーターの数として、直接解釈することができる。本論文では、以下の定理でこれを形式化する。：

定理 1 （ファイナリティ）――*善意のバリデーターが、所与の有効ブロックBについて、confidence 閾値 $t \geq f$でファイナリティに到達した場合、ブロックBと競合するブロックについて、いかなる善意のバリデーターも、confidence 閾値 tでファイナリティに到達することはない。*

前述で不可能とされた結果[DLS88]により、confidence 閾値が$\frac{n}{3}$以上の場合の liveness（生存性）を証明することは不可能である一方で、実際には大抵の場合、バリデーターの大半はプロトコルから逸脱することがないという点に留意したい。このような場合には、任意に高い confidence 閾値に到達することが可能であり、こう

[2] 大抵のシステムは、その動作に基づいてバリデーターにリワードや処罰を与える為、オフラインのノードや小規模な攻撃（ファイナライズされたブロックをリバートできないようなもの）は、活発な行動を極力制限されてしまい、ゆえに一般的なものとは考えられない。

[3] 実際、Highway においては、特定の閾値を選択すれば、バリデーターが他のバリデーターとの通信の出力に対して実行するローカルでの計算にのみ影響を与える。したがって、バリデーターがこうした通信ログを外部のオブザーバーに渡す場合、オブザーバーは異なる閾値を用いて同ログを再解釈することが可能な場合がある。

した期間に構築されたブロックをリバートすることは事実上不可能となる。

次に本論文では、プロトコルが、所与の confidence 閾値でブロックを確実にファイナライズし続けるのに必要な善意のバリデーターの数に関する限界値を提供する。本論文では同限界値が、cで表される「クラッシュ障害」の追加概念と共に、従来の限界値である$n \geq 3f + 1$に則したものであることに注目するが、このクラッシュ障害とは、コンセンサスの過程を阻害するものの、ビザンチン故障程の影響力はないものである。

定理 2 （Liveness）――*全ての confidence 閾値*$0 \leq t < \frac{n}{3}$*については、*$f \leq t$*であって、*$c < \frac{n-3t}{2}$*である場合には、confidence* t*でファイナライズされたブロックのチェーンは、善意のバリデーター各々について無制限に成長する。*

2.5 関連研究

部分的かつ同期的なプロトコルに関するこの分野の研究は、Dwork、Lynch、および Stockmeyer らが、その独創性に富んだ業績[DLS88]をもって開始し、PBFT[CL99]プロトコルとそれに関する多数のバージョン[BKM18, KAD+09, MNR19]が導入されることで好評を博した。従来型においては、このモデルにおけるプロトコルは、$n/3$未満の悪意あるノードに対して回復力を達成する。これは、ビザンチン故障の数値が高いと safety（安全性）と liveness（生存性）の双方を提供することは不可能であるとする既知の限界によるものである。[DLS88]しかしながら一部の研究は「flexibility（柔軟性）」の概念を探求しており、これは普通、通常の部分的かつ同期的なモデルにおける最強の$n/3$のセキュリティと、更には、ネットワークの同期性や限定的な敵対行動等の追加的な条件が満たされた場合に備えた追加保証とを提供することとして理解されている。

Gasper[BHK+20]は Ethereum 2.0 のコンセンサス・メカニズムに対して提示する新たな研究対象であるが、この Gasper が、ネットワークが同期性についての仮定を満たす場合に備える追加保証のケースに関して分析した。その後、非常によく似た検討事項が、プロトコルの snap-and-chat 系の導入で、適切かつ正式に研究された。[NTT20]この snap-and-chat のプロトコルは、2 つの「ファイナリティのレベル」を定義している。（同論文では 2 つの台帳として形式化されており、1 つの台帳が別の台帳の拡張という形になっている。）第一のレベルは、より高速で、従来型の部分的かつ同期的な仮定に依存しており、障害が$n/3$未満のノードに抑えられている限りは、liveness（生存性）と Safety（安全性）が確保されている。第二のレベルは、敵対するノードに対してより強い$n/2$の回復力を提供するものの、ネットワークが同期している間だけ liveness（生存性）を維持する。ネットワークの同期性がネットワークのレイテンシー（待ち時間）に関してかなり悲観的な仮定を必要とすると想定して実際に展開を行った場合と同様に、第二のレベルのファイナリティの進行はかなり遅いと仮定できる。提供する構造は非常にモジュール式である為、多種多様なプロトコルを使用して、第一と第二のファイナリティのレベルに対して備えることができ、双方のレベル間の一貫性が保証される。

到達度の高いセキュリティ保証について述べるならば、Highway との間に最も多くの類似性があると考えられるプロトコルは、Flexible（柔軟性のある）BFT であると言える。[MNR19]この Flexible BFT では、プロトコルの safety（安全性）に対する攻撃をうまく行えない場合に善意のノードと完全に協働する、不正ノード、すなわち、生存しつつも破損しているノードの新しいタイプについて定義している。直感的に言うと、Flexible BFT は、明示的インセンティブがないノードは不正を行わないという実際的な状況をモデル化している。これは PoS システムにおいては、不正を行えばリワードを解放してしまうことになる為である。Highway における状況と同様に、敵対するノードが単なる liveness（生存性）の破壊を意図せず、safety（安全性）の侵害のみに関心があるような場合には、同プロトコルは、非常に多くの敵対するノードを許容できてしまう。――この場合の限界は、本論文で示す限界と一致する。ファイナリティに関連するパラメーターの選択について述べるならば、Flexible BFT はその名が示す通り、ノードに対する一定の flexibility（柔軟性）をも導入していると言える。――各ノードは、各々の種類の不正ノードの数に関して独自の仮定を持つ場合があり、更に、正当な

（correct）仮定で成立している 2 個の善意のノードは、絶対に競合するブロックをファイナライズすることはできないということが保証されている。全てのノードが正当な仮定で成立している場合に、同プロトコルは機能し続けるのである。概念面での最大の相違点は、Flexible BFT では、ノードが保持する仮定が、プロトコル内の動作に明示的な影響を与えるのに対して、Highway では、仮定された confidence 閾値が、DAG（有向非巡回グラフ）上で実行されるローカル計算のみに影響を与え、この DAG の形式はバリデーターが行う特定の confidence 閾値の選択による影響を受けることはないという点である。こうした相違点から、主に 2 つの結果が生じることになる。第一には、Highway では、バリデーターは、他のバリデーターと通信せずとも、自分達の confidence 閾値を更新し、全てのブロックのファイナリティを再計算することができる一方で、Flexibility BFT の場合には、全プロトコルを再実行する必要がある。第二には、こちらの方がより重要と言えるかもしれないが、Flexible BFT では、善意のバリデーターの一部が不正を行う関係者の数に関して誤った仮定を行ってしまった場合に、全ての当事者の動作が失速してしまう場合があるということだ。これとは対照的に、Highway では、unit 生成が失速することは決してありえず、ある特定のバリデーターが、不正を行う関係者の数に関して誤った（incorrect）仮定を行った場合のみ、所与のバリデーターの見解からコンセンサスが失速する可能性はある。

この相違点を説明するのに、非常に保守的な善意のノードで構成された大きなグループが存在し、彼らがノードの 90%は善意的である、と誤って（incorrectly）仮定しているというシナリオを想定してみることにする。――このシナリオ内において、Flexible BFT では、あまり保守的ではない善意のノードがブロックをファイナライズすることができないのに対し、Highway では、あまり保守的ではない善意のバリデーターは、他のバリデーターのこうした選択による影響を受けることはない。これは、こうした選択が、お互いの通信に何ら影響を及ぼすことがない為である。――実際、バリデーターは、他者が選択する confidence 閾値を確認する明示的な手段さえ持っていない。

3 プロトコル

本論文では、ブロックチェーンの「ジェネシス・ブロック」をGで表し、これをプロトコル定義の一部と考える。ジェネシス・ブロック以外では、他の全てのブロックBは、$\mathrm{prev}(B)$で示す、ブロックBの親の参照、それから、通常はトランザクションのリストで示す、ブロックの内容で構成される。Bにおけるこの親参照は、Bにおいて$\mathrm{prev}(B)$のハッシュを含めることにより実現される為、ブロックのグラフでは循環が存在する可能性はない。本論文においては、ブロックBが親となる全てのブロックの集合についても$\mathrm{next}(B)$で表す。本論文では、ブロックの高さHを帰納的に定義する。：ジェネシス・ブロックの高さ$H(G)$は、0 であり、その他のブロックBについては、$H(B) = 1 + H(\mathrm{prev}(B))$となる。本論文では、ブロック$B_1$は、ブロック$B_2$の子孫であり、親リンクに従って（特に$H(B_2) \leq H(B_1)$となる場合に）、ブロック$B_2$からブロック$B_1$に到達する可能性がある場合には、$B_2 \leq B_1$と記述する。

3.1 DAG（有向非巡回グラフ）の構築

Highway プロトコルにおいては、提示されたブロックに関するコンセンサスに達することが目的で、バリデーターはメッセージを交換するのであり、それゆえに、生成されたブロックチェーンのうち、想定しうる限り多くの中の 1 つの分岐（branch）を検証（validate）する。既存のメッセージに関して異なるバリデーターが持つ知識を掌握し拡散する 1 つの方法として、DAG（有向非巡回グラフ）フレームワークが採用されており、

[Bai16, GLSS19, MMS99, ZRAP18]、このフレームワークでは、バリデーターが送信する全てのメッセージは、それ以前にバリデーターが送信したメッセージから成る特定の集合を参照する。本論文では、通常のプロトコル稼働中に送信されるこうしたメッセージを、*unit（ユニット、単位）*と呼び、それに含まれる参照を*citation（引用、引証）*と呼ぶ。より正式に表現すると、各 unit は以下のようなデータで構成される。

- **送信者**——unitの送信者（作成者）のID
- **Citation（引用、引証）**——作成者が証明することを希望する、他のユニットのハッシュのリスト
- **ブロック**——unitが、所与の時刻におけるブロック生成を指名されたバリデーターによって生成された場合には、同unitに含まれている

全ての unit が正当である（correct）と認められる為には、その送信者によるデジタル署名も必要とされる。本論文では、unit uの送信者（作成者）を$S(u)$と表現する。所与の unit が参照できるのは、それ以前に構築された unit のみであるので、unit に含まれる citation は、DAG（有向非巡回グラフ）のエッジ（edge）と考えることができる。

同プロトコルにおいては通常、所与の unit uが、ある他の unit u'を直接 cite（引用、引証）するのかどうかだけでなく、その unit 間に想定しうる因果関係が存在するのか、すなわち、unit uの作成者が、その作成中に、unit u'の存在を認識していたことを証明できるのかどうかといったことも重要になってくる。こういった概念は、unit uと unit u'を結びつけている citation のチェーンの存在によって容易に把握することができる。本論文では、こうしたチェーンの存在の下では、unit u'は unit uの *justification* であると言う。これを言い換えて、unit u'は unit uを justify する（証明する）とも表現できる。justification の関係は推移的である為、本論文においてはこれを、unit の集合における半順序として解釈し、$u' \leq u$によって、unit u'が unit uを justify するという事実を表す。本論文では、unit uよりも狭義で小さな全ての unit の集合をこの順序で示す。すなわち、$D(u)$で*下側単調集合*を、集合$D(u) \cup \{u\}$を$\bar{D}(u)$として示す。本論文では、自然な形で下側単調集合の表記を unit の集合に拡張する。すなわち、unit の集合 Sについては、以下のように定義する。

$$D(S) \stackrel{\text{def}}{=} \bigcup_{u \in S} D(u) \quad \text{さらに} \quad \bar{D}(S) \stackrel{\text{def}}{=} \bigcup_{u \in S} \bar{D}(u)$$

DAG フレームワークにおけるプロトコル・ビューの概念を形式化するのに、本論文では下記の定義を導入する。：

定義 1.（プロトコル・ステート）——*unit σの有限な集合が、Dの下で閉じている場合、すなわち、全ての$u \in \sigma$について、$D(u) \subseteq \sigma$である場合、unit σの有限な集合は、プロトコル・ステートである。*

新しい unit uを作成する場合には、バリデーターVは、その unit uより前に作成した中から最新の（最後の）unit を常に cite（引用、引証）する為、$D(u)$の中にある、それ以前に作成された unit を全て含んでいると想定できる。結果的には、善意のバリデーターによって作成された unit は常に、チェーンを形成する。悪意あるノードは依然として本ルールから逸脱する可能性があり、ゆえに以下のような定義となることに留意してほしい。

定義 2.（Equivocation）——*$S(u) = S(u')$が成立していて、unit uおよびu'が、$\leq$で示す比較が不可能な関係である場合、一対の unit (u, u')は、*equivocation *である。このような場合、送信者$S(u)$を*equivocator（equivocation 実行者）*と呼ぶ。*

本論文では、unit Sの所与の集合については、それで証明された equivocator の集合は以下のように示す。

$$E(S) \stackrel{\text{def}}{=} \{V \in \mathcal{V} \mid V \text{ が作成した} u, u' \in D(S) \text{が存在し、} u \not\leq u' \text{および} u' \not\leq u \text{ が条件 } \}$$

また（その時点まで）善意のバリデーターが unit uの下で生成した直近のメッセージの集合は以下のように示す。

$$L(u) \stackrel{\text{def}}{=} \{v \in D(u) \mid \text{全ての} v' \in D(u) \text{について、} S(v) \notin E(\{u\}) \text{および} v' > v \Rightarrow S(v') \neq S(v)\} \quad .$$

3.2 GHOST ルールによる vote（投票）

本節では、ブロックチェーンにおいてフォークを選択する際に適用する GHOST（Greedy Heaviest Observed Sub-Tree）ルールを導入し、DAG における仮想 voting（投票）と呼ばれる概念を用いて、その具体的な実装方法について説明する。

GHOST ルール——全てのブロックチェーン・クライアントが実行するタスクの中で重要なのが、フォークの選択である。必ずしも単一のチェーンを形成する訳ではないものの基本的にはブロックのツリーであるブロック$\mathcal{B}$を与えられた場合、その目的とは、そのブロックチェーンの head（頭）と考えられる「一端」（すなわち、このツリーの葉の1枚）を採取（摘み取る）することである。Bitcoin、またはEthereum等、プルーフ・オブ・ワークを基盤とするシステムにおいては、こうした目的を果たす為に最も一般的に使用されるのが、最長連鎖律（Longest Chain Rule）である。すなわち、最大の深さの葉（leaf）ブロックが、頭として選択される。

本論文の設定では、何らかの形で、フォーク選択ルールを用い、チェーンが「主要なもの」となっているバリデーターの committee の共通の信念を表現したいと考えている。とはいえこれは、バリデーターが「考えている」ことに関する追加情報がなければ不可能である。したがって本論文では、ブロック$\mathcal{B}$のほかに、バリデーターの opinion（意見）が与えられているものと想定してみる。このバリデーターの opinion は、マッピングによって以下のように表され、

$$\text{opinion} : \mathcal{V} \to \mathcal{B},$$

ここでは、バリデーター$V \in \mathcal{V}$の観点からして、ブロックのopinion(V)がブロックチェーンの頭であるはずだと考えられる。このブロック・ツリーおよびopinionといった要素が整えば、GHOST ルールは以下のように定義可能となる。

ブロック・ツリー$\mathcal{B}$と opinion 関数についての GHOST ルール

1. 各$B \in \mathcal{B}$については、total(B)が、バリデーター$V \in \mathcal{V}$の合計値となるよう計算する。ただし、opinion$(V) \geq B$の条件（すなわち、opinion(V)は、Bの子孫であるという条件）を満たすこととする。
2. Bをジェネシス・ブロックと設定する。Bが$\mathcal{B}$内の葉（leaf）でない間は、次のステップを繰り返す。：
   (a) total(B')の最大値を伴う$B' \in \text{next}(B)$を選択する。（B'のハッシュを用いた端数処理を行う）
   (b) $B := B'$と設定する。
3. Bを出力する。

簡潔に表現する為、本論文においては、GHOST$(\mathcal{B}, \text{opinion})$と記し、これを、opinion$: \mathcal{V} \to \mathcal{B}$を用いて GHOST ルールをブロック$\mathcal{B}$のツリーに適用して生じるブロックを表すものとする。

DAG（有向非巡回グラフ）を利用した仮想 voting（投票）——GHOST ルールは非常に自然かつ直感的であり、——このすぐ後の説明で示すように——数多くの望ましい特性を有する。ただし、このルールを適用するノードについては、ブロックのツリーと、各バリデーターが行う「最新の」vote（投票）が必要となる。問題となるのは、ノードはブロック・ツリーに関するローカル・ビューをいかに維持すべきか、更には、他のノードに関し、現時点での意見について何を考慮すべきかという点になってくる。現時点での意見の内容に関する 2 つのノード間の不整合がわずかなものであっても、それが原因で、GHOST ルールが、異なるブロックチェーンの頭を出力する可能性もあることに留意する。さらに悪いケースでは、不正なノードによって、こうした不整合が意図的に生成される場合があり、それは、様々なopinionを別々のバリデーターに送信したり、opinionを選択的に送信したりするといった形で行われる。

Highway においては、ノードのローカル・ビュー間の整合性は、DAG の力を借りて達成される。大まかに言えば、各 unit uは、$D(u)$のみに依存する仮想 GHOST の vote（投票、投票権）を有し、場合によっては、unit uの中には（存在する場合には）ブロックが含まれる。この vote（投票）は、unit uの下での「最新のメッセージ」における、仮想 GHOST の vote によって自動的に決定される。簡潔に表現する為、本論文では下記のように、バリデーターVが存在するとした場合に、そのVによって形成された固有の unit $v \in L(u)$を$L_V(u)$と示し、それ以外の場合を$L_V(u) = \perp$と示す。unit uが$\mathrm{vote}(u)$と見なすものを正式に定義するのに、まずは、「uに対してローカルである」opinion 関数、$\mathrm{opinion}_u : \mathcal{V} \to \mathcal{B}_u$を定義する。ここで、$\mathcal{B}_u$は$\overline{D}(u)$にある全てのブロックによって構成されている為、unit uは、それが有するブロックに対して vote を行う場合もある。：

$$\mathrm{opinion}_u(V) \stackrel{\text{def}}{=} \begin{cases} \mathrm{vote}\left(L_V(u)\right) & \text{if } L_V(u) \neq \perp \text{ の場合} \\ G & \text{その他の場合} \end{cases}$$

さらにこれに続いて、本論文では$\mathrm{vote}(u)$を以下のように定義する。

$$\mathrm{vote}(u) \stackrel{\text{def}}{=} \mathrm{GHOST}(\mathcal{B}_u, \mathrm{opinion}_u).$$

これは、プロトコルの正当性(correctness)に影響を及ぼすことはない一方で、善意のバリデーターが、unit uにおいて提示されているブロックBが$\mathrm{vote}(u) = B$の条件を満たしているかを常に確認する場合には、効率性の面で最善であることを意味する。これは、ブロックBの親を$\mathrm{GHOST}(\mathcal{B}_u \setminus \{B\}, \mathrm{opinion}_u)$として選択することで達成することが可能であり、unit uの正当性に関する必要条件となることも可能である。

$\mathrm{vote}(u)$に関しては、unit uが作成された瞬間にその unit uの作成者が、ブロックチェーンの頭と見なしているブロックとして解釈することもできる。これは、unit uの作成者が、$\mathrm{vote}(u)$がいずれファイナライズされるであろうと確信していることを意味している訳ではない。その代わりに、各バリデーターは、所与のブロックが「リバート」される、すなわち、最終的にブロックチェーンの一部とはならない可能性を示す confidence（信頼性）パラメーターを、ブロック毎に維持している。本論文の次節にて説明するが、この confidence パラメーターの値は、所与のブロックをリバートする為に、unit に対して equivocation を行う必要のあるバリデーターの数と比例している。

3.3 ファイナリティの条件

DAG と vote（投票）のメカニズムについての定義を行ってきたが、ここでいよいよ Highway プロトコルにおけるブロックをファイナライズする際のルールを導入していく。DAG フレームワークを利用することにより、バリデーターは、ローカルでの操作のみを実行しながら、つまり、DAG のローカルコピーにおける特定の構造を検索しながら、各ブロックのファイナリティを計算することが可能である。

定義 3 *プロトコル・ステートσに比例する、ブロックBに関する$(q,k) - summit$は、unit $(C_0, C_1, C_2, \ldots, C_k)$の集*

合から成るネスト化された数列である。ただし、$C_0 \supseteq C_1 \supseteq \cdots \supseteq C_k$と以下が条件となる。

- **全会一致 (unanimity)** 全ての$u \in C_0$について、$\mathrm{vote}(u) \geq B$
- **善意性 (honesty)** $E(\sigma) \cap S(C_0) = \emptyset$,
- **コンベクシティ (convexity)** 全ての$u_0 \leq u_1 \leq u_2$ついて、$u_0, u_2 \in C_i$は$u_1 \in C_i$ を意味する。ただし、$S(u_1) = S(u_2) = S(u_3)$が条件となる。
- **密度 (density)** 全ての$u \in C_{i+1}$について、$|S(\bar{D}(u) \cap C_i')| \geq q$ となる。

ただし、 $C_i' = \{u \in C_i | u' \in C_{i+1}$が存在し、$\mathrm{S}(u) = \mathrm{S}(u')$が条件$\}$

直感的な話をすると、summit は連続する複数の unit を表し、この unit は、同一のブロックBへの vote を行うバリデーターから成る大きな部分集合（すなわちクォーラム）が生成したものである。また DAG の構造として、その後で vote が変更される可能性が非常に低い為、summit がブロックのファイナライズに利用されるのである。特に、本論文においてはこれから一連の補題で証明していくことになるが、バリデーターは、以下の条件を確認すれば、tを超える数のバリデーターが equivocation を行わないと所与のブロックBのリトラクトとはならないという事実を容易に納得できると思われる。：

$$\mathrm{FINAL}(B, \sigma, t) \stackrel{\mathrm{def}}{=} \begin{cases} 1 & \sigma\text{において、ブロック Bに関する}(q,k)-\text{summit が存在する場合、} \\ 0 & \text{それ以外の場合} \end{cases} \quad (2q-n)(1-2^{-k}) > t \text{ が条件となる}$$

ここで上記の定義に基づいて、confidence 閾値tが整数値である場合に、バリデーターVを、t*に関してファイナライズされた*ブロックBを有するものとする。この際に、$\mathrm{FINAL}(B, \sigma, t) = 1$と、$\sigma$がバリデーター$V$のプロトコル・ステートであることが条件となる。次に、一定の高さとクォーラム・サイズを有する summit が発生した後では、新しい unit がその summit の vote に反対票を投じることはないという事実を示しながら、重要な技術的な補題を証明する。

補題 1 *$C = (C_0, \ldots, C_k)$を、プロトコル・ステート$\bar{D}(C_0)$に比例する、ブロックBに関する$(q,k)-summit$とし、$\sigma \supseteq \bar{D}(C_0)$を、任意の正当な（correct）プロトコル・ステートとし、unit uを、σにおける 1 つの unit とする。ただしここでは、$D(u) \cap C_k \neq \emptyset$と、$vote(u) \not\geq B$を条件とし、よって、$f \geq 2(q-n)(1-2^{-k})$となる。*

証明. 本論文では以下のように定義する。

$$A(u) = E(u) \cup \left(S\left(\{ v \in L(u) \mid \mathrm{vote}(v) \not\geq B \} \right) \cap E\left(C_0 \cup D(u) \right) \right)$$

本論文においては、少し強い命題、すなわち、補題の条件が満たされている場合に$|A(u)| \geq 2(q-n)(1-2^{-k})$となることを証明する。

ここでは、kに関する帰納法で進めていく。$k = 0$の場合は、$2(q-n)(1-2^{-0}) = 0$いった形で、普通に論を進めていくことができる点に注目する。よって、これは$k-1$についても保持されると仮定する。

補題の仮定を満たす unit uが存在すると仮定する。すなわちここでは、$D(u) \cap C_k \neq \emptyset$と、$\mathrm{vote}(u) \not\geq \mathcal{B}$が条件となる。本論文では、この$u$の最小値を取る。
$u_k \in D(u) \cap C_k$とし、さらに$\mathcal{V}_{k-1} = S(\{v \in C'_{k-1} | v < u\})$とする。unit uは、C_kにおいてある unit の上方にある為、$|\mathcal{V}_{k-1}| \geq q$となることは明らかである。$\mathcal{V}_{k-1}$の全てのバリデーターが、ブロック$B$か、それより上方にあるブロックに対する vote をし続ける場合、unit uも同様のことを行い、それゆえに、その一部は equivocation を行ったか、vote を変更したはずであると考えられる。加えて、$\mathcal{V}^{\mathrm{eq}}_{k-1} = \mathcal{V}_{k-1} \cap E(u)$とし、さらに、$\mathcal{V}^{\mathrm{change}}_{k-1} = \mathcal{V}_{k-1} \cap S(\{v \in L(u) | \mathrm{vote}(v) \not\geq B\})$と定義する。unit uは、ブロック B に対して賛成の$q - |\mathcal{V}^{\mathrm{eq}}_{k-1}| - \left|\mathcal{V}^{\mathrm{change}}_{k-1}\right|$以上の vote（投票）についても、さらには反対の vote についても、その上方にある為、以下を得ることができる。[4]

$$q - |\mathcal{V}^{\mathrm{eq}}_{k-1}| - |\mathcal{V}^{\mathrm{change}}_{k-1}| \leq \frac{n - |E(u)|}{2} \leq \frac{n - |\mathcal{V}^{\mathrm{eq}}_{k-1}|}{2}$$

$$2q - n \leq |\mathcal{V}^{\mathrm{eq}}_{k-1}| + 2|\mathcal{V}^{\mathrm{change}}_{k-1}| \tag{1}$$

$\mathcal{V}^{\mathrm{change}}_{k-1}$が空である場合、本論文では既に命題を得ていることとなり、$\mathcal{V}^{\mathrm{change}}_{k-1} \neq \emptyset$と仮定する。これにより、$u'$を最小の unit とし、ここでは$u' \leq u$と、$\mathrm{vote}(u') \not\geq B$と、さらに、$D(u') \cap C_{k-1} \neq \emptyset$を条件とするが、これは$\mathcal{V}^{\mathrm{change}}_{k-1}$が非空である為に必ず必要な条件と言える。帰納法の仮定により、本論文では以下を得る。

$$|A(u')| \geq (2q - n)(1 - 2^{-k+1}) \tag{2}$$

次に、不等式 1 と 2 を接続することにより、以下を得ることができる。

$$|A(u') \cup \mathcal{V}^{\mathrm{eq}}_{k-1}| + |\mathcal{V}^{\mathrm{change}}_{k-1}| \geq$$
$$\frac{|A(u')|}{2} + \frac{|\mathcal{V}^{\mathrm{eq}}_{k-1}| + 2|\mathcal{V}^{\mathrm{change}}_{k-1}|}{2} \geq$$
$$\frac{(2q - n)(1 - 2^{-k+1})}{2} + \frac{2q - n}{2} = (2q - n)(1 - 2^{-k})$$

各$V \in \mathcal{V}^{\mathrm{change}}_{k-1}$が、$C_k$において unit を生成する必要があったこと、さらに、ブロックBと競合するブロックへの vote を証明する unit も生成する必要があったことに注目する。後者の unit が、C_kにおいて unit の上方にある場合は、unit uの最小性と矛盾してしまう為、バリデーターVは$E\left(C_0 \cup D(u)\right)$に属している必要があり、結果として、$A(u)$にも属している必要がある。本論文においては、 $A(u') \subseteq A(u)$、さらに$\mathcal{V}^{\mathrm{eq}}_{k-1} \cap \mathcal{V}^{\mathrm{change}}_{k-1} = \emptyset$ともしている為、集合$A(u')$および$\mathcal{V}^{\mathrm{change}}_{k-1}$が互いに素である点も示しておく。

[4] unit uがカウントする vote（投票）合計値が、$n - |E(u)|$となる点に留意のこと。

$V \in A(u') \cap \mathcal{V}_{k-1}^{\text{change}}$が存在するとする。$\mathcal{V}_{k-1}^{\text{change}} \cap E(u) \neq \emptyset$である為、$V \in E\big(C_0 \cup D(u')\big)$となる必要があり、さらに、狭義で$u'$の下にある最後の$V$の unit は、ブロック$B$と競合している、あるブロック$B'$に vote（投票）を行わなければならない。$v$をブロック$B'$に vote を行うこの unit とし、$v' \in C'_{k-1}$を$\mathcal{V}_{k-1}$に存在するバリデーター$V$の証明者とする。バリデーター$V$は、$u'$による equivocator（equivocation 実行者）とは見なされない為、本論文では、$v' \leq v$を得ることができ、これにより$v < u'$の時に$\text{vote}(v) \not\geq B$、$D(v) \cap C_{k-1} \neq \emptyset$と$v \leq u$が得られるが、これは$u'$の最小性と矛盾する。

補題 2（ファイナリティ）——*V_1とV_2を、その各々のプロトコル・ステートであるσ_1とσ_2を有する善意のバリデーターとし、B_1とB_2を、競合する 2 つのブロックとすると、*$\text{FINAL}(B_1, \sigma_1, t_1) = 1$、$\text{FINAL}(B_2, \sigma_2, t_2) = 1$*の場合には、$f > \min(t_1, t_2)$となる必要がある。*

証明. $\sigma = \sigma_1 \cup \sigma_2 \cup \{u_{max}\}$とし、ここで$u_{max}$は、善意のノードの 1 つが作成した追加 unit であり、そのノードは下側単調集合として完全な$\sigma_1 \cup \sigma_2$を有することが条件である。σは、プロトコル実行の際に必ず出現する人為的なステートというわけではないものの、依然として正当な（correct）ステートであるということができ、それゆえに補題 1 をこれに適用しているという点に留意する。本論文ではFINALの定義により、σ_1に比例するブロックB_1に関しては、$(q_1, k_1) - \text{summit}$を得、$\sigma_2$に比例するブロック$B_2$に関しては、$(q_2, k_2) - \text{summit}$を得る。本論文では定義上、以下を得る。：

$$(2q_1 - n)(1 - 2^{-k_1}) > t_1 \tag{3}$$

$$(2q_2 - n)(1 - 2^{-k_2}) > t_2 \tag{4}$$

σがプロトコル実行において到達できない場合であっても、補題 1 は、σと同様に、正当なステートを保持するという点に留意する。加えて、summitの定義により、あるプロトコル・ステートσ'に対応している任意の$\text{summit}(C_0, \ldots, C_k)$は、プロトコル・ステート$\overline{D}(C_0)$に対応しているsummitでもある。$u_{max}$は、こうした summit 双方の上方にある為、補題 1 から、$\text{vote}(u_{max}) \not\geq B_1$である場合に、$f \geq (2q_1 - n)(1 - 2^{-k_1}) > t_1$であり、さらに、$\text{vote}(u_{max}) \not\geq B_2$である場合には、$f \geq (2q_2 - n)(1 - 2^{-k_2}) > t_2$を得る。ブロック$B_1$とブロック$B_2$は競合する為、$\text{vote}(u_{max})$は、そのどちらの上方にもあり得ず、それゆえに本論文では、$f > \min(t_1, t_2)$を得ることができる。

最後に、単純系として定理 1 を証明する。

証明. 2 つのバリデーターが、競合するブロックBとB'に関して、confidence 閾値$t \geq f$でファイナリティに到達したという矛盾を仮定する。ただしその後で本論文では、補題 2 により、$f > \min(t, t) = t$を得る為、これをもって本証明は完結する。

3.4 ファイナリティ条件の計算可能性

本節においては、所与のステートσおよびクォーラム・パラメーターqについて、可能な限り高い$(q, \cdot) - \text{summit}$の相対$\sigma$を得る為に、単純で貪欲なアルゴリズムを利用することが可能であることを示していく。まずは、本アルゴリズムの疑似コードから対応していく。

アルゴリズムSUMMIT(σ, B, q)

1. $S_0 \subseteq \mathcal{V}$を、$\sigma$においてequivocationを行っておらず、その最新のメッセージがブロックBへのvote（投票）を行った、全てのバリデーターの集合とする。

2. C_0を、S_0によって作成されたunitの集合とし、これは以下のように構築されたものであるとする。：各バリデーター$V \in S_0$について、

   (a) uを、σにおいてバリデーターVが作成した最新のunitとする。
   (b) uが、ブロックBへのvoteを行う時に：
   - uを、C_0に加える。
   - uを、uの下側単調集合においてバリデーターVが作成した uの直接の親に置き換える。

3. $l = 1, 2, 3, \ldots$について以下を繰り返す：

   (a) $S_l := S_{l-1}$を設定する。
   (b) 以下を繰り返す：
   i. $W := S_l$を設定する。
   ii. $C_w := \{u \in C_{l-1} : S(u) \in W\}$とする。
   iii. 各$V \in W$について
   • $|D(u) \cap C_w| \geq q$と共にあるバリデーターVによる$u \in C_{l-1}$が存在しない場合、S_lからVを除く。
   iv. $W = S_l$の場合、ループを終了（break）する。

   (c) $S_l = \emptyset$の場合、$\{C_0, C_1, \ldots, C_{l-1}\}$を返す（return）。

   (d) $C'_{l-1} := \{u \in C_{l-1} : S(u) \in S_l\}$とする。

   (e) $C_l := \{u \in C_{l-1} : |D(u) \cap C'_{l-1}| \geq q, S(u) \in S_l\}$とする。

以下の補題は、SUMMITアルゴリズムが常に有効な$(q, \cdot) - \text{summit}$を出力し、さらに、返される（returned）summitが最大であることを示している。

補題 3 （SUMMIT の特性）——$(C_0, C_1, \ldots, C_r)$を$\text{SUMMIT}(\sigma, B, q)$*の出力とし、これにより以下となる。*

1. **（Correctness、正当性）** $r \geq 1$*の場合、*$(C_0, C_1, \ldots, C_r)$*は、*σ*に比例するブロック*B*に関する*$(q, r) - \text{summit}$*である。*
2. **（Maximality、最大性）** σ*に比例するブロック*B*に関する、*$(q, k) - summit(D_0, D_1, \ldots, D_k)$*の全てについては、*$i = 0, 1, \ldots, k$*の各々に対して、*$r \geq k$*と、*$D_i \subseteq C_i$*を得る。*

証明. **Correctness（正当性）**——C_0がコンベクシティを満たし、$i = 0, 2, \ldots, r,$については、全てのC_iの要素がC_0の部分集合として善意性（honesty）および全会一致（unanimity）を満たしているということは、アルゴリズムのステップ 1 と 2 を見れば明白である。$i > 0$の場合の、C_iに関する密度は、アルゴリズムのステップ 3.(e)において保証されており、C_iをこのように構築することによりコンベクシティをも意味している。

さらに証明する必要があるのが、全てのC_iの要素は非空であるという点である。本論文においては帰納的に進めていく。：C_0は、$r \geq 1$である為、非空である。ここで、$1 < i \leq r$の場合に、$C_{i-1} \neq \emptyset$であるとし、本論文では$C_i \neq \emptyset$を示す。$r \geq i$は、アルゴリズムのステップ 3.(c)においては、$S_i \neq \emptyset$を意味する為であることに注意す

る。さらには、3.(c)に到達した後では、各バリデーター$V \in S_k$が$|D(u) \cap C'_{i-1}| \geq q$を満たす 1 つ以上の unit uを作成したことを、3.(b)のループが保証している。これにより、C_iは非空であるということになる。

最適性――本論文では、帰納法によりさらに論を進め、$i = 0, 1, \ldots, k$の各々について、$D_i \subseteq C_i$さらに、$S(D_i) \subseteq S(C_i)$となることを示していく。summit のコンベクシティ要件と、アルゴリズムのステップ 1 および 2 により、基礎ケース（base case）$i = 0$が保持される。

ここで、$i = 0, 1, 2,$の場合に、$D_i \subseteq C_i$および $S(D_i) \subseteq S(C_i)$が保持され、ある$l \leq k$の場合に$l - 1$が保持されるとする。本論文では、$D_l \subseteq C_l$と$S(D_l) \subseteq S(C_l)$を示す。この目的の為に、まずは以下の不変式を示す。：反復lの間、アルゴリズムのループ 3.(b)に示される集合Wが、$S(D_l) \subseteq W$を満たす。最初の反復では、$W = S_{l-1}$が得られ、$S(D_l) \subseteq S(D_{l-1})$であり、さらに帰納$S(D_{l-1}) \subseteq S(C_{l-1}) = S_{l-1}$がある為、この不変式が保持される。ここで、ループ 3.(b)のステップ iii において、Wから 1 つのバリデーター$V \in W$を取り除く場合はいつでも、D_{l-1}内にあるVによる unit uはいずれも、$|D(u) \cap C_{k-1}| \geq q$を満たさない。$S(D_{l-1}) \subseteq W$である為、これにより、$V$のいずれの unit も、$|D(u) \cap D'_{l-1}| \geq q$を、また特に$V \notin S(D_l)$を満たさないということになる。

言い換えれば、集合Wから 1 つのバリデーターVを取り除く場合はいつでも、同バリデーターは$S(D_l)$に属することはできず、それゆえに依然として$S(D_l) \subseteq W \setminus \{V\}$となる。結果としてこの不変式が保持され、その代わり、それにより、$S(D_l) \subseteq S_l = S(C_l)$ということになる。$D_l \subseteq C_l$が含まれているのは、単に$S(D_l) \subseteq S(C_l)$であることから導くことができる結果であり、$C_l$に関連する部分に関してはアルゴリズムのステップ 3.(e)において構築されている。
単純な結果として、ファイナリティ基準FINAL(B, σ, t)は、以下の通り、効率的に計算することが可能である。

FINAL(σ, B, t)の**計算**

1. $q = \left\lceil \frac{n+t}{2} \right\rceil, \left\lceil \frac{n+t}{2} \right\rceil + 1, \ldots, n$については、以下のようにする：

   (a) kを、SUMMIT(σ, B, q)が出力したSummitの高さとする。
   (b) $(2q - n)(1 - 2^{-k}) > t$*の場合は*、1を**返す（return）**。

2. 0を**返す（return）**。

3.5 Liveness（生存性）の保証

これまで本論文においては、プロトコル実行中に生成される DAG の構造特性について論じてきており、一定の組み合わせ構造――summit――が DAG 内にある場合に、これをブロックのファイナリティを完了するのに使用できることを示した。とはいえ、こういった組み合わせ構造が実際に DAG に出現すること、さらに敵対者は任意の進歩への妨害を無制限に行うことができないということについての証明を依然として行っていない。こうした証明に取り組むにあたり、本論文ではまず、バリデーターが新しい unit をいかに、そしていつ生成すべきか、そしていつ新たなブロックをその中に入れていくのかを示すストラテジーを定義する。

unit 作成スケジュール――本論文が、部分的かつ同期的な通信モデルについて取り組んでいることを想起してほしい。このモデル内においてはある値$\Delta > 0$が既知であり、ここでは GST と呼ばれるある未知の時点以後、ある善意のバリデーターからの各メッセージが受信者に届くまでにかかる時間がΔ未満であることを条件とする。本論文では、バリデーターは完璧にクロックを同期させているという標準的な仮定を行う。（代わりに、現地時間における各々の限界付きの時間差を、メッセージ配信における遅延として扱うこともできる。）

本論文においては、プロトコル実行を、長さ $R := 3\Delta$の複数のラウンド（round）$r = 0, 1, 2, \ldots$に分割する。各バリデーターは、クロックに基づいて、ローカルでラウンドを追跡する。1 個のラウンド・リーダー（round leader）$\mathrm{LEADER}(r) \in \mathcal{V}$を、1 個のラウンド指数（round index）$r \in \mathbb{N}$に対して割り当てる、1 個のリーダー・スケジュール（leader schedule）$\mathrm{LEADER} : \mathbb{N} \longrightarrow \mathcal{V}$が存在すると仮定する。複数のラウンド・リーダーは、新しいブロックを作成する責任を負っており、それゆえに、複数の善意のリーダーがこのスケジュール内に可能な限り頻繁に現れることを確認することが重要であるということは想像がつくだろう。理論的には、本論文において仮定する必要があるのは、善意のバリデーターが何度も無限に現れるということだけである。実際には、例えば、ラウンドロビン（round-robin）・スケジュールか、または疑似ランダム生成スケジュールを使用することができる。

以上をもってここで、善意のバリデーターが、新しい unit を作成する際にいかに動作すべきかについてのストラテジーを導入してもよいであろう。

ラウンド$r \in \mathbb{N}$における$V \in \mathcal{V}$による unit 作成および受信ストラテジー

/* 時間は、ラウンドrの開始から計測する。
/*バリデーターVが新しい unit を作成する場合にはいつでも、ローカル DAG における全ての unit の最大値の上方にあり、DAG にすぐに追加される。

1. 時刻$t = 0$においては、$V = \mathrm{LEADER}(r)$の場合には、
   (a) 全てのunitを、*バッファ（buffer）*からローカルDAGに移動する。
   (b) 新しいunit u_L（**proposal（提案）unit**と呼ぶ）を作成し送信する。新しいブロックBをuに入れ、その親を選択して、$\mathrm{vote}(u_L) = B$となるようにする。

2. タイムスロット$(0, R/3)$内において、$V \neq \mathrm{LEADER}(r)$である場合：$\mathrm{LEADER}(r)$が作成した新たなunit u_Lを受信した場合には、すぐにそれを(その下側単調集合と共に)ローカルDAGに追加する。u_Lを追加した直後に、新しいunit（このラウンドにおいてはVによる**confirmation（確認）unit**と呼ぶ）を作成し送信する。同スロットにおいて受信した残りの全てのunitはバッファに置く。

3. 時刻$R/3$においては、unitを、バッファからローカルDAGに移動する。

4. タイムスロット$(R/3, 2R/3)$内においては、:新しいunitを受信した場合はいつでも、すぐにそれをローカルDAGに追加する。

5. 時刻$t = 2R/3$においては、新しいunit（このラウンドにおいてはVによる**witness（証明者）unit**と呼ぶ）を作成し送信する。

6. タイムスロット$(2R/3, R)$内においては：同スロットで受信した全てのunitはバッファに置く。

バリデーターが confirmation unit の作成をスキップする場合（時刻$R/3$以前に leader から unit を受信していない場合）があることに注意すること。一方で、witness unit は、いつでも作成される。したがって、全てのラウンドにおいて、ラウンド・リーダーは 2 個の unit――proposal unit と witness unit を作成し、leader 以外は、1 個（witness unit）か、2 個（confirmation unit と witness unit）の unit のいずれかを作成する。

補題 4 *$t \geq f$を、ある整数の confidence パラメーターとし、クラッシュ障害性のあるノードの数を$c < \frac{n-3t}{2}$と仮定する。善意のバリデーターが、ステートσに到達した場合、あるブロックBについてのFINAL(σ, B, t)であることが条件となり、次に、ある時点以降で、任意の善意のバリデーターが到達する全てのステートσ'については、FINAL(σ', B, t)が保持される。*

証明. ビザンチン故障性もクラッシュ障害性もない全てのバリデーターの集合をHで示す。定義上では、nとtは整数である為、本論文では、$c \leq \frac{n-3t-1}{2}$を得ることができ、それゆえに以下となる。

$$|H| = n - f - c \geq n - t - \left(\frac{n-3t-1}{2}\right) = \frac{n+t+1}{2}. \tag{5}$$

まず、GST 以後、各バリデーター$V \in H$は最終的に、完全にσを含むステートに到達する点に留意する。この状態となった際には、r_0を最初のラウンドとし、補題 1 により、これらのバリデーターのいずれかがこれ以降に作成した各 unit uは、vote$(u) \geq B$を満たすことになる。

本論文では、バリデーター$V \in H$がラウンドrにおいて作成した witness（証明者） unit（すなわち、時刻$2R/3$に作成されたもの）を、u_r^Vで示す。$r \geq r_0 + 1$については、本論文では、$u \geq u_r^V$と共にVが作成した任意の$u \in \sigma$によって満たされる以下の特性を得ることができる。

1. $W \in H$について、uは、全てのunit u_{r-1}^Wを、その下側単調集合に含む。
2. vote$(u) \geq B$

前者の特性は GST 以後にr_0の状態が発生したという事実から成立し、後者は補題 1 の結果である。：確かに、特定の summit の存在により、ファイナリティに到達する。——同 summit を超えて構築される各 unit uはその後、ブロック$\geq B$への vote（投票）を行う必要があり、それゆえに、vote$(u) \geq B$となる。ここで、複数の units $\{u_r^V\}$のファミリーを使用して、任意に高い複数の summit を構築することができることが容易に理解できる。もっと具体的に言えば、あるステートσを与えられたとして、以下の複数の集合を定義する：

$$i \geq 0 \text{ について} \quad C_i := \left\{u \in \sigma : S(u) \in H, u \geq u_{r_0+i}^{S(u)}\right\}$$

したがって、集合C_iは、バリデーター$V \in H$の各々に関する、$u_{r_0+i}^V$で始まる unit のチェーンの和集合である。本論文では、以下の場合はいつでも、$(C_0, C_1, \ldots, C_k)$は、プロトコル・ステートσに比例する$(|H|, k) - \text{summit}$であることを主張する。

$$|S(C_k)| = |H|, \tag{6}$$

このことは、上にリスト化した 2 つの特性からもすぐに当然のものと捉えられる。さらには、Hにおける全てのバリデーターは善意的である為、H内の各バリデーターは、上記 (6) が保持されている、ステートσに最終的に到達することになる。

さらに、上記 (5) により、$k > \log(t+1)$については、$|H| \geq \frac{n+t+1}{2}$である為、

$$(2|H| - n)(1 - 2^{-k}) \geq (t+1)(1 - 2^{-k}) > t$$

が得られることにも留意する。

結果として、$\log(n)$のラウンドの後では、善意のバリデーターの各々のステートσ'について、FINAL(B, σ', t)が保持される。

定理 2 の証明：

証明. 補題 4 により、単一のバリデーターであっても、あるブロックのファイナライズが行われる場合にはいつでも、最終的に、他の全てのバリデーターがそれに続き、全く同じブロックをファイナライズしていくことになるという点に注意する。したがって、高さ$h \geq 0$では、あるブロックB_hがファイナライズされるが、高さ$h+1$ではどのブロックに対しても(任意の善意のバリデーターによる)ファイナライズが行われないというようなプロトコル実行が存在する矛盾も想定しておかなくてはならない。

ビザンチン故障性とクラッシュ障害性のどちらも有していないバリデーターの全ての集合をHで示す。本論文では、補題 4 で証明されているように、$|H| \geq \frac{n+t+1}{2}$を得ている。r_0をラウンド指数とし、以下をその条件とする：

1. GST以後に、ラウンド$r_0 - 1$となっていること。
2. ラウンド$r_0 - 1$が開始する前に、すでに各バリデーター$V \in H$が、B_hをファイナライズし、さらにその結果として、その1つ前のラウンド内の全てのunitにおいて、B_hの上方でブロックにvoteを行ってしまっていること。
3. Leader $V_L := \text{LEADER}(r_0)$が$H$のメンバーであること。

このようなラウンドr_0の存在については、補題 1 の内容からも（B_hをファイナライズする Summit 上に構築された各 unit は、ブロック$\geq B_h$への vote を行う。）、さらに、H内のバリデーターがスケジュール内に際限なく頻繁に出現するという事実からも、当然のことと捉えられる。ラウンドr_0でリーダー V_Lによって提示されるブロック$\mathcal{B}$が最終的にファイナライズされる（したがって、高さ$h+1$には絶対に達しないという事実との矛盾にいたる）ということも示していく。

この目的を果たす為に、ラウンドr_0の開始時点でのV_Lのステートをσ_0で示し、ラウンドr_0の開始時点でV_Lが作成する proposal（提案）unit をu_0で示し、さらに、V_Lがu_0において提示するブロックをBで示す。V_Lは、Bの親として GHOST の選択を行う為、本論文では$\text{vote}(u) = B$が得られる。さらに、H内のバリデーターによる直近の全ての unit（$\text{round}(r_0 - 1)$において作成されたもの）は、B_hの上方でブロックへの vote を行う為、$B_h \leq B$となる。ここで、任意のバリデーター$V \in H$について考慮し、u_Vを、ラウンドr_0の間に、Vによって作成された中の最初の unit（すなわち、Vがそのラウンドのリーダーである場合は proposal unit であり、それ以外の場合には、時刻$R/3$よりも前に作成された、Vの confirmation（確認）unit）とする。本論文では以下を主張する。

$$D(u_V) = \sigma_0 \cup \bar{D}(u_0).$$

このことは、GST が既に経過してしまっているという事実から考えても、unit 作成および受信ストラテジーから考えても成立する。実際、Vによりその 1 つ前の unit $\bar{u}_V$が作成された、ラウンド$r_0 - 1$内の時刻（time）$2R/3$と、Vがu_0を受信した、ラウンドr_0内の瞬間（moment）$< R/3$との時間の間に、($R/3 = \Delta$である為に、Vが時刻$R/3$より前にu_0を受信することは保証されている)、Vは、u_0とその下側単調集合以外の他の unit をその DAG に含めていなかった。さらにV_Lは、u_0を作成する前に$\bar{u}_V$を受信しており、したがって、$\bar{u}_V \in D(u_0)$となるため、この主張が成立するのである。

この主張と、GHOST ルールと、さらに$|H| > n/2$という事実から、$\text{vote}(u_V) = B$を得ることができる。さらにはu'_Vが、このラウンドにおいてVによる（時刻$2R/3$に作成された）witness（証明者）unit である場合に、$\text{vote}(u'_V) \geq B$も得られるが、これは全ての$W \in H$について [5]$u_W \in L(u'_V)$となる為である。（GST は既に経過してしまっている為、u'_Vが作成される前に、Vがこうした各u_Wを受信することが分かる。）同様の論法により、ラウンド$(r_0 + 1)$で善意のバリデーターが作成する（すなわち、時刻$2R/3$に作成する）各 witness unit u''_Vは、

[5] 本論文において、より強力なバージョンである$\text{vote}(u'_V) = B$ではなく$\text{vote}(u'_V) \geq B$を主張する理由は、不正なバリデーターが途中で、Bの上で他のブロックを提示した可能性がある為である。

その下側単調集合において全ての unit $\{u'_V\}_{W\in H}$を、さらに$\text{vote}(u''_V) \geq B$を有していると言える。この引数を繰り返していくことにより（補題 4 の証明における類似の引数を参照のこと）、全ての$k > 0$について、ラウンド$r_0 + k$の開始によって、善意のバリデーターは各々、プロトコル・ステート σ'に比例する$(|H|, k) - \text{summit}$ を含むステート σ'に至るとの結論を得ることができる。

なお、補題 4 において見られる計算により、$k > \log(t+1)$については、本論文では、こうした summit が既にブロックBをファイナライズしていることが分かっている。結果として、ラウンド$r_0 + k$以後は、善意のバリデーターの各々のステートσ'については、$\text{FINAL}(B, \sigma', t)$が保持される。これにより、高さ$\geq h+1$のファイナライズされたブロックが得られる為、いかなる高さ$h+1$のブロックもファイナライズされないという仮定との矛盾に至ることとなる。

3.6 通信複雑度

本節にいたるまでに説明してきたプロトコルは、通信複雑度に関して言えば、ビザンチン故障性のあるノードが存在しない状況においては、かなり効率的である。実際、各バリデーターは、平均でラウンドあたり 2 個の unit を生成し、通信し、同一のラウンドで他のバリデーターが作成した$\approx 2n$個の unit をダウンロードする。あるバリデーターがビザンチン故障性を有していて、通信複雑度を増大させようとし、場合によっては、善意のバリデーターのリソースを使い果たし、クラッシュを引き起こしてしまうといった場合には、状況ははるかに困難なものとなる。その一つの可能性として考えられるのが、無効なメッセージや不正な形式のメッセージを善意のバリデーターに大量に送信するというものである――こういった攻撃は危険な場合がある一方で、こうした不正な（incorrect）受信メッセージの全てを、単に破棄するだけでも対処することは可能だ。もう 1 つの可能性としては、プロトコル・ルールに従って生成されていないのに、依然として正当な（correct）メッセージのように見える為に、善意のノードが破棄することができないメッセージすなわち unit を送信してしまうというものだ。結局のところ、後者のタイプの攻撃に対しては、実装段階においてではなく、プロトコル内で対処済みとなっている必要があるが、これは、そうした対処をしなければ、プロトコルの通信複雑度は原則的に際限がなくなってしまう為である。不正なバリデーターが仕掛けてくる、この種の基本的な*スパム・パターン*については、以下の 2 つに区別することができる。：

- **垂直（vertical）スパム**――単一の不正なバリデーターが、equivocationを行わないものの、速いペースで新しいunitを生成し「unit作成および受信ストラテジー」に違反する。この種のスパムは、所与の時点までにバリデーターが生成するはずのunit数を正確に計算することが可能である為、取り扱いが容易だ。また、その受信数が増加する場合でも、単に無視することができる。

- **水平（horizontal）スパム**――単一の不正なバリデーターか、あるいは不正なバリデーターのグループが、多数のequivocationを生成し、それらを異なる善意のバリデーターと並行して送信し、DAGを大きくする。定義によれば、これらがequivocationとなり、これにより攻撃者は処罰を受ける場合がある（ステークが削減される可能性が高い）が、それにもかかわらず問題を起こすことがある。

後者のタイプの攻撃は、はるかに危険で対処が難しいものとなる。こういった攻撃の中で最も基本的な形式においては、単一の悪意あるバリデーターが、equivocation を行う vote（投票）を、他のバリデーターに対して非常に大量に送信する。他のバリデーターが、自分のローカル DAG に、これら全ての vote を追加してしまうと、自分のパソコンで利用可能な全メモリをすぐに使い果たしてしまうことになる。この種の直接的な攻撃に関しては、バリデーターに対し、自分達の下側単調集合において、equivocation を行う vote を直接 cite（引用、引証）するのを拒否させることにより、非常に容易に回避できる。（本論文では後程、この考え方に関するもう少し洗練されたバージョンについて論じる。）これにより、**攻撃者がただ 1 人**存在する場合に、その攻撃者

から送信されたメッセージのチェーンを、善意のバリデーターの各々が直接 cite する数は最大でも 1 つまでという状態を確保することができる。これはつまり、equivocation を行う単一の攻撃者は、その unit を$\Omega(n)$個（善意のバリデーター毎に 1 個）生成することができるということを意味する。こうした動作は厳格に処罰される為、この制限値は合理的なものであると言え、したがって、仮にこのような事態が発生したとしても稀なことであると思われる。

とはいえ、**多数の攻撃者**が結託している場合には、上記のような防御では有効な解決策とはならない為、ここで話を終えるのは妥当ではない。攻撃者であるバリデーター$A_1, A_2, B_1, B_2, C_1, C_2, \ldots \in \mathcal{V}$が、結託して下記のようなひとまとまりのメッセージを作成しているとする。

1. A_1は複数のunit m_1を作成し、A_2は 単一のunit m_2を作成する。
2. m_1は、unit m_{11}とm_{12}を、ブロックB_1とB_2の各々により、直接Cite（引用、引証）する。
3. m_2は、equivocationを行うunits m_{21}とm_{22}を、ブロックB_1とB_2の各々により再び、直接citeする。
4. unit m_{ij}は、C_1とC_2によりm_{ij1}とm_{ij2}の上方にある。

このパターンを繰り返しつつ、$2k$個のバリデーターが共謀することで、k回のラウンドの過程上に2^k個の vote から成る 1 つのパターンを作成することができ、結果として、どの vote も equivocation を直接 cite することはない。任意の善意のバリデーターが、このパターンにおける最上位の unit の検証（validate）を試みる場合、その最下位にあって equivocation を行う unit を、指数関数的に大量にダウンロードすることが必要となる。おそらくここでさらに事態を困難にしているのは、これらの unit を受信する善意のバリデーターにとって、何らかの不正が起こっていることを伝えるのは容易であるものの（桁外れの量の unit や equivocation を与えられた場合に）、その不正についての責任者が誰なのか、不正が発生している unit としてどの部分を切り離すべきなのかを正確に決断するのは一筋縄ではないという事実である。例えば、上述のパターンにおいては、A_1とA_2は実際には equivocation を行っておらず、したがって善意のバリデーターと見分けがつかない可能性がある。つまり、こうしたパターンを受信しているバリデーターVは、これらのメッセージを無視するか（すなわちA_1とA_2が善意のバリデーターであった場合に、これらのバリデーターは互いに恒久的に分離していくことになる）、あるいはこれらのメッセージを転送してしまうか（このバージョンが、別の equivocation を行うメッセージと共に、他の善意のバリデーターへと送信された場合には問題となる可能性がある）という選択肢を押し付けられたまま放置されることになる。上記の攻撃については、*Fork 爆弾*と呼び、[GLSS19]において説明してきた。

unit は他の unit を cite するものであり、そうした他の unit は既知の unit と equivocation を行うものであるが、攻撃への対処にあたっては、こういった unit に対して特段、疑い深くなる必要がある。特に、これらの vote がそれ自体は equivocation ではないということを理解していきたい。本論文においては、endorsement（承認）のシステムを導入することにより、これを解決する。

3.6.1 Endorsement（承認）を利用したスパム防止

equivocation を行う単一のバリデーター$A \in \mathcal{V}$は、その他の$n-1$個のバリデーターの各々に対して、equivocation を行う unit を送信する可能性があり、次の下側単調集合に含める unit についての調整を行わない場合には、その全てが最後には DAG の一部になってしまう可能性があり、したがって、善意のバリデーターの各々は、Aからの$\Omega(n)$個の unit のチェーンを含めて相互の unit に組み込まなければならないということに注意する。したがって、バリデーター間で、DAG に全ての単一の unit を含めるかどうかの調整を希望しないならば、（equivocation を行う場合に）バリデーター毎にチェーンを$\Omega(n)$個というのが、本論文で望める最善であ

るという事実を受け入れなくてはならない。

本論文では、楽観的なケース――つまり DAG において equivocation が全く存在しないというケース――における、調整を全く必要としないストラテジーを一つ提示する。そして悲観的なケース――つまり equivocation が存在するというケース―においては、あるオーバーヘッドを加えるだけである。それと同時に、これは望みうる最高の保証を達成する。：善意のバリデーターのローカル DAG には、equivocator（equivocation 実行者）毎で$\Omega(n)$個を超える unit のチェーンが含まれることはないのである。この目的を達成する為に、本論文では、プロトコルに新しいタイプのメッセージである endorsement を導入する。

1 個の endorsement（承認）$\mathrm{ENDORSE}(V,u)$は、1 個の unit uに関して、1 個のバリデーター$V \in \mathcal{V}$が（バリデーターVがデジタル署名する）送信する 1 個のメッセージである。直感的な表現をすれば、このメッセージの意味は以下のように示すことができる。：

$$\mathrm{ENDORSE}(V,u) \equiv \textit{このendorsementを作成する際、}V\textit{は}S(u)\textit{によるequivocationを認識していなかった。}$$

本論文では、$> n/2$個のバリデーターVがuの endorsement を送信した場合に、unit は承認済みである（endorsed）と表現する。承認されるということは、バリデーターのローカル・ビューに依存する為、一種の主観的特性であるという点に留意する。

$V \in \mathcal{V}$に関する Endorsement（承認）ストラテジー

1. *緩和（relaxed）*モードにおいて、バリデーター$\mathcal{V}$がプロトコルを開始する。
2. 任意のequivocationが行われた後に、バリデーターVが*警戒（cautious）*モードへと切り替わる。[a]
3. *警戒モード*に入る際には、バリデーターVが認識している全てのunit uについては、Vが$S(u)$による任意のequivocationを認識していない場合には、Vは $\mathrm{ENDORSE}(V,u)$を送信する。
4. *警戒モード*にある間、バリデーターVが初めてunit uについて学習しており、$S(u)$による任意のequivocationを認識していない場合はいつでも、Vは $\mathrm{ENDORSE}(V,u)$を送信する。
5. 任意のモードにおいて：相異なるバリデーターWから、unit uに関する、endorsementの数が$> n/2$の$\mathrm{ENDORSE}(W,u)$が届いた場合はいつでも、uのステータスを*承認済み（endorsed）*として設定する。

[a] この警戒モードの明白な終了条件は存在しない。ただし、本論文の小節4.2において規定したように、プロトコル実行は複数のeraに分割され、各eraの開始時点では、全てのバリデーターは緩和モードで初期化される。

全ての「善意の」unit に対して endorsement を送信する為、endorsement の送信に関する上記のストラテジーは、かなりナイーブ（naive）で世間知らずとも言えるものであると言えることを、本論文では強調しておく。このバージョンは、一種の簡易解析を考慮したものであるが、実用面においては若干効率的ではない可能性がある。第 4 節において、洗練されたストラテジーを示し、同様の目的を達成しているが、ここでは送信する endorsement をより少ないものとしている。

本論文においては、善意のバリデーターは、equivocation を行うペアの unit を endorse（承認）することはないということに注意する。これは、これらの unit の第 2 番目のものを受信する時までに、その送信者が equivocator であることに気付いてしまう為、endorse することはないからである。実際に、単一のバリデーターが equivocation を行うメッセージの endorsement を提示することは、同バリデーターの不正な動作を検証できる証拠にもなり、それによって同バリデーターにペナルティを与えることもできる。さらに、これまでと同様に、$f < \frac{n}{3}$の不正なバリデーターが存在すると仮定し、同時に endorse される同一のバリデーターによる、無

比の unit の最大数が 3 であると証明できる。本論文では大抵の場合、endorse される unit は、かなりの割合の善意のバリデーターによって認証されたものと考える。他の unit の下側単調集合が疑わしい可能性があると見なされる一方で（equivocation 爆弾を含んでいる可能性があるとして）、endorse された unit はより信頼性がある（solid）と見なされる。

定義 4 *任意の unit uは、ナイーブではあるものの（naively）、もう 1 つの unit vを cite（引用、引証）するものとし、$u > v$であって、endorse された unit wが存在しない場合に、これを$u >_n v$で示し、ここで$u > w \geq v$をその条件とする。*
このナイーブな citation の概念を用いて、本論文においては、unit に関する検証（validity）の基準を導入するが、これにより equivocation 爆弾を回避することができる。

Limited Naivety Criterion（LNC）——限定的な単純基準

（あるバリデーターWによって作成された）equivocation (v_1, v_2)と、Vが作成した 2 つの unit $u_1, u_2 \leq u$が存在する場合に、バリデーターVによる unit uは、不正（incorrect）であるものと見なされる。ただし、$u_1 >_n v_1$および$u_2 >_n v_2$をその条件とする。

以前にも指摘した通り、endorse された unit と、そしてそれゆえにナイーブと言える citation は主観的な概念である為、所与の unit によって、LNC（Limited Naivety Criterion、限定的な単純基準）をも満たすことは、バリデーターのローカル・ビューに依存する可能性がある。重要なのは、所与のバリデーターについての任意の unit のステータスが、ただ 1 つ、変更する場合があるという点である。——所与の unit が LNC を満たす場合には、それを満たして終了することはなく、任意の unit が endorse される場合には、endorse されて終了することはなく、さらには、任意の unit がもう 1 つの unit vによってナイーブな citation を受けない場合には、vによるナイーブな citation を受けるとは見なされないのである。したがって、unit 受信ストラテジーを適切に修正して、所与のバリデーターが依然として LNC から外れずにその上に自分の unit を構築できていない unit を、バッファ内に残したままにしておくことは重要である。いったん、こうした unit が十分な endorsement を集めてしまえば、バッファから取り除くことができ、DAG に加えることができる。[6].

おそらく、「限定的な単純基準（Limited Naivety Criterion）」の最も重要な特性は、任意のメッセージの下側単調集合に含まれる、equivocation を行う vote の数が多すぎないということが示せる点である。もっと具体的な説明をすると、以下の定理で述べる通り、unit の数を、善意の unit よりも下に制限することができる。

定理 3. *$f < \frac{n}{3}$と仮定する。uがある善意のバリデーターが作成した N番目の unit であり、uが最大でも $O\left(nN(1 + f_{\text{equiv}})\right)$の、他の unit の上方にある場合には、ここでは$f_{\text{equiv}} = |E(D(u))|$が条件となる。*

以下の小節においては、定理 3 の証明について示す。

3.6.2 Equivocation 数の制限

ここではまず、一度に承認できる独立した equivocation の数を制限しながら、単純な補題に取り組むことにする。

[6] バッファの無制限の成長を抑制する為、本論文では、小節 4.2 にて定義するように、各 era の最後でこれを取り除く。

補題 5 $f < \frac{n}{3}$*として、プロトコル・ステート*σ*を、プロトコル実行中の、任意の善意のバリデーターのステートとする。*$C \subseteq \sigma$*を、あるバリデーター*W*が作成した、ペアで無比の unit の集合とする。*C*における全ての unit が endorse（承認）される場合、*$|C| \leq 3$*となる。*

証明. 各 unit $u \in C$について、uを endorse（承認）する善意のバリデーターの集合を、$E_u \subseteq \mathcal{V}$で表す。このような各$u$は仮定によって endorse される為、本論文では、$|E_u| + f < n / 2$が得られる。

善意のバリデーターは、単一のバリデーターからの 2 つの無比の unit を承認しない為、複数の集合E_uは互いに素であるということになり、ゆえに

$$n - f \geq \sum_{u \in C} |E_u| > |C|(n/2 - f),$$

となり、さらに、$f < \frac{n}{3}$である為、$|C| < 4$ということになる。

定理 3 の証明は、以下の技術面に関する補題に依存しており、同補題は、任意の不正なバリデーターが作成することができる異なる equivocation の数を制限し、これにより、その全てが善意のバリデーターのステートに含まれることになる。

補題 6 $f < \frac{n}{3}$*と仮定する。プロトコル・ステート*σ*を、任意の善意のバリデーターのプロトコル・ステートとする。（したがって、プロトコル・ステート*σ*内の全ての unit は、「限定的な単純基準（Limited Naivety Criterion）」を満たす。）次に、任意の unit* $u \in \sigma$*について、さらには任意のバリデーター*V*について、*$D(u)$*において*V*が作成した unit の集合は、最大で*$O(n)$*個の unit のチェーンの和集合に含まれる。*

証明. Cを、uの下側単調集合において単一のバリデーターVが作成した、ペアで無比の unit の任意の集合とする。本論文ではこれを$|C| \leq 3f + (n - f) + 1$と表す。これを得ることにより、Dilworth の定理から判断して、この補題が成立する。

第一に、最大で 1 個の unit $u_0 \in C$が存在する可能性があり、ここでは、$S(u)$によるある unit が、u_0をナイーブに cite（引用、引証）することが条件となる。（このことは、LNC を満たすuから判断しても成立する。）明確に説明する為、本論文においては、Cからこのような任意のu_0を取り除き、その後で、得られた上限に$+1$を加算することにより、これについて考察していく。この時点では、各$v \in C$については、$v \leq e \leq u$を条件として、1 つの endorse された unit eが存在する。各$v \in C$については、本論文では、e_vを選択して、endorse された全ての unit eの集合における、任意の最小値の unit とし、ここでは、$v \leq e \leq u$を条件とする。

本論文においては以下を主張する：

- $v \in C$に関する複数のunit e_vは、相異なるものである。
- $|\{e_v : v \in C\}| \leq 3f + (n - f)$

これらの主張を確立すれば、当然の結果として同補題が成立するということは、容易に理解できる。

第一の主張については、$v_1 \neq v_2$と共に、$v_1, v_2 \in C$について、$e_{v_1} = e_{v_2}$が得られた場合には、e_{v_1}は、2 つの equivocation をナイーブに cite し（このことは、e_{v_1}の最小性から判断して成立する。）、したがって LNC に反する可能性がある。

第二の主張については、本論文ではまず、ある$v_1, v_2 \in C$について、$S(e_{v_1}) = S(e_{v_2})$である場合、e_{v_1}, e_{v_2}は無比となるはずであることに注目する。実際、一般性を失うことなく、$e_{v_2} \leq e_{v_1}$が得られた場合には、$v_2 <_n e_{v_2}$および$v_1 <_n e_{v_1}$である為、e_{v_1}は LNC に反する可能性がある。つまり、補題 5 により、各 equivocator

（equivocation 実行者）は、最大で 3 個の unit を$\{e_v : v \in C\}$に対して寄与する場合があり、残りのバリデーターは、最大で 1 個の unit を寄与する場合がある。これは全体で、$|\{e_v : v \in C\}| \leq 3f + (n - f)$になる。
補題 6 は、equivocation スパムを用いて作成されうるメッセージ数の限界を提供する。このことは十分、定理 3 の証明となる。

定理 3 の証明 uの作成者は善意的であり、未来からの unit をその DAG に加えたりしない為、$D(u)$において単一のバリデーターが作成した unit のチェーンの各々は、最大でも$O(N)$の長さしかない可能性がある。補題 6) により、$D(u)$内の各 equivocator は、最大で$O(n)$個のチェーン、その各々について$O(N)$個の unit を寄与する可能性がある。したがって、$D(u)$内の equivocator による unit の数は、最大で$O(nNf_{\text{equiv}})$個ということになる。残りの善意のバリデーターは、これらのN回のラウンドの間に、最大で、合計$O(nN)$個の unit を生成する。

3.6.3 スパム防止策追加後の Liveness（生存性）

「限定的な単純基準（Limited Naivety Criterion）」を適用すれば、攻撃者が善意のノードに対し、過剰な数の unit 処理を強いることができない状況を確保できる一方で、本論文で述べている元々の liveness（生存性）ストラテジーでは、バリデーターが認識している全ての unit の上方に unit を作成する必要があった為、これでは LNC に違反する可能性があり、そうなると、同ストラテジーをもって動作させることができるかどうかはもはや明確であるとは言えない。したがって、この元々の liveness ストラテジーに若干の修正を行う必要がある。

まず、ラウンド長さを$R = 3\Delta$から$R = 6\Delta$まで伸ばす。その理由としては、本論文においては、時刻 $0, R / 3$、または$2R / 3$に善意のバリデーターが送信する unit に関しては、同 unit のみでなく、その endorsement も、時刻$R / 3$、$2R / 3$、またはRのそれぞれの時間までに他の善意のバリデーターの各々に届くようにしたいからである。さらに本論文では、バリデーターが *不審*モードにある場合はいつでも、同バリデーターに対して直接 endorse（承認）された unit のみを cite（引用、引証）するように指導する。（バリデーターの直近の unit はこのルールを免除される。）

主要な liveness の結果についての証明――定理 2―を成立させる為、以下の要件が満たされるようにする必要がある。

Rapid Endorsement Spread（急速 Endorsement）：「GST 以後に善意のバリデーターVが unit uを作成する場合はいつでも、時刻$R / 3$以後は、uの作成時刻にVのローカル・ビューにおいて endorse された$D(u)$内の各 unit は、全ての善意のバリデーターのローカル・ビューにおいても endorse される。」

上記の条件を保証するのに最も容易な（とはいえおそらく最も効率的とは言えない）方法は、特定の unit がその善意のバリデーターのローカル・ビューにおいて endorse された後に、その unit の$> n / 2$個の endorsement の集合を送信するように、そのバリデーターに対して指導するということになるだろう。ここではもっと効率的なストラテジーが色々と考えうるが、本論文では第 4 節において、endorsement に対するもっと実用的なアプローチを論じていく。

4 実用面での考慮事項

4.1 動的なラウンド長さ

本論文で述べるプロトコルの基本的なバージョンを定義するにあたり、時間を複数の固定長のラウンドに分割する。ラウンド長さの選択は、そうした各ラウンドにおいて、最悪のケースによる遅延を想定しつつも、善意のバリデーター間での少数の通信である「ラウンド・トリップ」発生の可能性を確認することを目的として行われた。理論的な側面から述べると、最悪のケースによるメッセージの遅延Δは、当初[DLS88]において定義した通り、*既知の*$\Delta flavour$における部分的かつ同期的な特性に関する仮定から得ることは可能であった。実際問題としては、実験を用いて、適切なΔを見積もることはできるが、定数のΔを設定するにあたっては認識しておくべき少数の危険が存在する。：

- Δを楽観的すぎる値（すなわち低すぎる値）で設定した場合には、（高いpingの）一対のバリデーターping間のメッセージがあまりに遅く届く場合がある為、liveness（生存性）が脅かされる可能性がある。

- Δを悲観的すぎる値で設定した場合には、ラウンドのうち長期間において、バリデーターが停止状態となってしまう為、プロトコルがかなり遅くなる可能性がある。

このことは、動的なラウンド長さを用いる動機づけになり、プロトコル自体もその時点でのネットワーク条件に適合していく。理論的なレイヤーでは、この「アプローチ」は、[DLS88]による部分的かつ同期的な性質に関する当初の定義における*未知の*$\Delta flavour$に対応している。理論的なモデルでは、ある固定された、最悪のケースであるメッセージの遅延Δが存在するとされているが、プロトコル実行の開始において、それを得るのは不可能である。これら2つの部分的に同期的な性質に関するモデルが、ほぼブラックボックス還元を介して、同等のものであったとしても、本論文ではここで、実装面に最適化した未知のΔモデル向けの、このプロトコルのカスタム・バージョンを提示する。

ラウンド長さを変更するルール——本論文では、Unixエポック以来のミリ秒にて時間を計測する。[7]整数のタイムスタンプは$tick$と呼ばれる。本論文ではこれまでと同様に、一般性を失うことなく、参加者が同期クロックを持っていると仮定する。本論文ではバリデーター$\mathcal{L}(i) \in \mathcal{V}$の1つを、各tick iに対して、tickの*リーダー*として疑似ランダムに割り当てる。善意のバリデーターがこのスケジュールに頻繁に出現するように、同スケジュールが十分にランダムであることを確認する必要がある。

本論文ではunitがタイムスタンプを有すると仮定する。すなわちunit uに関して、$T(u)$は、unit uが送信された時刻にあたる。

各バリデーターvは、定期的に更新されるプライベートなパラメーター$n_v(i) \in \mathbb{N}$（*ラウンド指数*と呼ぶ）を維持する。正式に表現すると、マップ $n_v : \mathbb{N} \longrightarrow \mathbb{N}$は、バリデーター$v \in \mathcal{V}$がこの特定のtickで用いるべきラウンド指数を、各tickの数に対して割り当てる。本論文においては、n_vの選択方法に関するストラテジーを下に示していくが、iが、$2^{n_v(i)}$と$2^{n_v(i-1)}$の両方の倍数でない場合に、常に$n_v(i) = n_v(i-1)$であると仮定する。言い換えれば、jから$j + 2^{n_v(i)} - 1$までの間、$2^{n_v(i)}$のtickのタイムウィンドウに対して、同パラメーターは一定に保たれると仮定するが、ここでは、$j \leq i$は最大であり、$2^{n_v(i)}$を除算することが条件となる。

バリデーター$v \in \mathcal{V}$に関しては、最新の tick iが$2^{n_v(i)}$で整除でき、このラウンド長さが$R := 2^{n_v(i)}$である場合はいつでも、新しいラウンドが開始する。さらには、全ての tick iについて、値$i \bmod 2^{n_v(i)}$は、ラウンドが開始してから経過した時間を決定する。このラウンドについては、このバリデーターは、小節 3.5 において説明されている unit 受信および作成ストラテジーに従う。このラウンドのリーダーは、$\mathcal{L}(i_0)$であり、ただしここではi_0は、同ラウンドが開始する時点での tick である。

本論文においては（以下で説明する通り）、n_vと、ゆえにラウンド長さも、各バリデーター$v \in \mathcal{V}$のそれぞれに

[7] Unix 時間：1970 年（UTC：協定世界時）の開始からの秒数であるが、本項の目的においてはミリ秒の解像度がより適切である。

ついて、そのローカル・ビューのみに基づいて変更されるということを強調する。こうした理由により、ラウンドは、善意のバリデーター間においてでさえ、時折同期がずれてしまう可能性がある。（そして確実にずれてしまう。）とはいえ、ラウンド長さを調節するストラテジーはこのように設計されているのであり、結果として、実際にこのようなことが起こるのは稀であって、ラウンド長さは 2 の倍数によってのみ違いが出るということになる。

n_vを変更するストラテジー——本論文のストラテジーは、以下の数によってパラメーター化されている。：

- それぞれが最小値と最大値を決定する2つの自然数$n_{\min} < n_{\max}$、バリデーターを考慮に入れたn_vの値
- ファイナライゼーションの進行度を測定するのに利用する、confidence閾値t_0
- ラウンド指数の変化の基礎となる条件を規定する数、$0 < C_{\text{fail}} < C_{\text{succ}} < C$および$D$

$V \in \mathcal{V}$に関するラウンド指数を維持する為のストラテジー

1. tick $i = 0$では、全てのjについて、$n_v(j) = n_{\min}$を初期化する。$\text{cnt}_{\text{succ}} = 0$を初期化する。
2. tick $i > 0$では、$m := n_v(i)$とする。iが、2^{m+1}で整除できる場合はいつでも以下の通りとなる。

   (a) b_{fin}は、最後のCのラウンドの間、confidence t_0でファイナライズしたブロックの数を示す。
   (b) $b_{\text{fin}} \leq C_{\text{fail}}$の場合は、全ての$j \geq i$について、$n_v(j) = \min(m+1,\ n_{\max})$を設定する。
   (c) $b_{\text{fin}} \geq C_{\text{succ}}$の場合は、$\text{cnt}_{\text{succ}} = \text{cnt}_{\text{succ}} + 1$をインクリメントし、それ以外の場合には、$\text{cnt}_{\text{succ}} = 0$を設定する。
   (d) iが、$C2^{m+1}$で整除でき、$\text{cnt}_{\text{succ}} \geq D$である場合、全ての$j \geq i$について、$n_v(j) = \max(m-1,\ n_{\min})$を設定する。$\text{cnt}_{\text{succ}} = 0$を設定する。

実際問題として、t_0はおそらく、「safety（安全性）」の為に利用する confidence 閾値よりもはるかに低いはずであり、例えば、$t_0 \approx 0.01$である。単一のラウンド内であっても、リーダーが提示するブロックは、より低い confidencet_0でファイナライズされ、十分な善意のバリデーターが存在する場合は、summit の高さの値が増加するのに従い、将来的にこの confidence も増加する。「ファイナリティ値」のフィードバックに基づいてバリデーターが「正当な（correct）」ラウンド指数を学習しているのを確認する為に、定数C_{fail}と、C_{succ}と、Cと、そしてDが選ばれる。これらの定数の値の例を挙げると、$(C_{\text{fail}}, C_{\text{succ}}, C) = (10, 32, 40)$および$D = 3$となる。

4.2 Era

Highway プロトコルはブロックチェーンの作成と維持を目的としたものである為、一旦実行（初期化）されれば、永久に停止することなく動作していくとされる。結果的には、バリデーターは DAG 全体を、プロトコル実行のごく初期に作成された unit でさえも、強制的に保存（store）させられることになる。DAG の「古い（初期の頃の）」部分を消去することは安全ではなく、これは、（不正である可能性が高い）バリデーターがその直近の unit において、そうした非常に古い unit を直接に cite（引用、引証）する可能性がある為である。

この保存、つまりストレージ（storage）に関する問題について論じる際には、ブロックチェーンの保存は何らかの方法で行っていかなくてはならず、これにかかるコストは不可避であるということを、明確な形で強調していくことが重要である。ただし、直感的な話をすれば、古い「メタデータ」すなわち一定のブロックをファイナライズするのに利用され、もはや使い道のない unit の消去を止めるべきではない。大規模なバリデーターの committee にとって、この「メタデータ」には、ブロックチェーン自体よりももっと多くの記憶領域を必要とする可能性が高いということが分かると、この問題はかえって深刻なものとなる。更に悪いことに、こうし

た DAG はディスクではなく RAM に保存しなくてはならない。その理由は、ファイナリティを検出したり、（特に LNC において）unit の正当性（correctness）を確認したりする目的で、DAG 上に効率的なデータ構造を構築しなくてはならないからである。代わりにディスク上にこうしたデータ構造を配置してしまうと、処理速度が劇的に落ちてしまう。

この問題に対する実用的な解決策は、プロトコル実行を複数の *era* に分割することである。各 era においては、$K = 1000$個の新たなブロックがブロックチェーンに追加されるが、重要なのは、Highway の新たなインスタンスは、全ての era において実行されるということである。つまり例をあげると、本論文では era 5 においては、ジェネシス unit として era 4 でファイナライズされた、高さ 4999 のブロックB_{4999}について考察するということになる。

本論文にてこのような方法で era を利用する場合、バリデーターは、その直前の複数の era においてファイナライズされたブロックと、直近の era の DAG を保存しさえすればよいことになる。特定の era でとらわれた equivocator（equivocation 実行者）は、後続する era のかなり最初の段階でアクセス禁止をされる場合がある為、このことは、equivocation スパム攻撃に対しても大変有効である。善意のノードは、*緩和（relaxed）* モードで新しい era を開始する為、過去の equivocation に起因する endorsement（承認）をもはや送信する必要はない。

Era を使用することで、さらに追加できるメリットとして挙げられるのが、前もって指定したあるルールに従ってバリデーターの集合を変更することが可能であり、それゆえに同プロトコルを自由参加型モデルへと移行することができるという点である。

4.3 より少ない Endorsement を送信

本論文の小節 3.6.1 において導入した当初の endorsement ストラテジーはとても簡潔なものであったことを想起してほしい。：equivocation について考察した後で、equivocation を行わないバリデーターによる全ての unit を endorse（承認）する。このストラテジーにより、Liveness（生存性）が保持される非常に単純な引数を考慮していく一方で、equivocation が特定の era において検出された場合に、軽微でないオーバーヘッドも導入する。

本論文では、下記の洗練されたストラテジーを提示するが、このストラテジーを用いれば、善意のバリデーターが、その善意の unit の endorsement（承認）の不足で立ち往生するといった事態にはならないと依然として保証できる。同時に、同ストラテジーは、はるかに少ない endorsement を送信することが可能である。

$V \in \mathcal{V}$に関する洗練された Endorsement（承認）ストラテジー

1. 集合$Equivocators := \emptyset$を初期化する。

2. バリデーターVが、Wによるequivocationの実行に対応する場合はいつでも、Wを$Equivocators := \emptyset$に加える。

3. バリデーターVが認識している全てのunit uに関しては、以下の条件を満たしていることとし、Vは、$\text{ENDORSE}(V, u)$を送信する。：

   (a) uの作成者である$S(u)$は、$Equivocators$には属さない。
   (b) バリデーター$W \in Equivocators$が存在し、以下がその条件となる。

- $W \notin E(u)$
- Wによって作成されたunit wが存在し、$w \in D(u)$がその条件となるが、$S(u)$によって作成され、uのすぐ前にあるunit u'は、wをCite（引用、引証）しない。

実用化におけるシナリオにおいては、バリデーターWによる equivocation がラウンドrにおいて検出される後では、数回のラウンドの後、いずれの善意のバリデーターも、もはやWによる任意の新たな unit を cite（引用、引証）することはなくなる。上記のストラテジーにおいては、このようなケースの場合、ラウンドrの後に発生する、この数回のラウンドにおいて作成される unit の為だけの endorsement（承認）を送信することが必要となる。後で作成される善意の unit は、Wによる任意の「新しい」unit を cite することはなく、したがって、endorsement（承認）を必要としない。

4.4 Weight を加味したコンセンサス

これまでに、全てのバリデーターの opinion が、コンセンサスに到達する過程において全て同様に重要であるということを仮定してきた。本小節においては、各バリデーターがその voting （投票）権に対応している、関連する整数の weight（重さ）を持つというシナリオにおいて、Highway プロトコル実行を可能にする修正――これは例えば、プルーフ・オブ・ステークのブロックチェーンの構築の際に有用なバージョンである――について説明する。

全てのバリデーターの weight（重さ）の総和はNに等しいとし、関数$\mathrm{w}(V)$は、バリデーターVと関連する weight を示す。こうしたバージョンにおいては、関数 total は、GHOST ルールの基盤（basis）であり、その数の代わりに、所与の選択値をサポートするバリデーターの weight の*合計値*を数える関数と置換する必要がある。同様に、summit の定義において示した**密度（density）**の条件については、その数の代わりに、バリデーター$S\big(\bar{D}(u) \cap C_i'\big)$の集合の weight の合計値を制限するバージョンと置換する必要がある。これらの変更を導入した後でも、定理 1 と 2 は何の変更をしなくても真となるが、これは、これらの定理がバリデーターの数に対応しておらず、この証明は同じバリデーターの weight を利用しない為である。

こうした修正をおこなったシナリオにおけるファイナリティは純粋に、バリデーターの weight の一関数である一方で、その通信複雑度はバリデーターの合計*数*に依存したままであり――全てのバリデーターは、その weight がどれ程小さい値であっても、全ての他のバリデーターの unit をダウンロードする必要があることに注意する。この理由から、ブロックチェーン・システムの設計にあたっては、バリデーターの数は、その weight や各々の間の依存関係ではなく、レイテンシー（待ち時間）や処理情報量に影響を及ぼす主要な要素であると考えるべきなのである。

参考

[AMN+19] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. Sync hotstuff: Synchronous smr with 2δ latency and optimistic responsiveness. *IACR Cryptology ePrint Archive*, 2019:270, 2019.

[Bai16] Leemon Baird. The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance. *Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep.*, 2016.

[BG17] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *CoRR*, abs/1710.09437, 2017.

[BHK+20] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining ghost and casper. *CoRR*, abs/2003.03052, 2020.

[BKM18] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. *CoRR*, abs/1807.04938, 2018.

[CL99] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In Margo I. Seltzer and Paul J. Leach, editors, *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*, pages 173–186. USENIX Association, 1999.

[CS20] Benjamin Y. Chan and Elaine Shi. Streamlet: Textbook streamlined blockchains. In *AFT '20: 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020*, pages 1–11. ACM, 2020.

[DLS88] Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.

[GAG+19] Guy Golan-Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K. Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. SBFT: A scalable and decentralized trust infrastructure. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019, Portland, OR, USA, June 24-27, 2019*, pages 568–580. IEEE, 2019.

[GLSS19] Adam Gagol, Damian Lesniak, Damian Straszak, and Michal Swietek. Aleph: Efficient atomic broadcast in asynchronous networks with byzantine nodes. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019, Zurich, Switzerland, October 21-23, 2019*, pages 214–228. ACM, 2019.

[KAD+09] Ramakrishna Kotla, Lorenzo Alvisi, Michael Dahlin, Allen Clement, and Edmund L. Wong. Zyzzyva: Speculative byzantine fault tolerance. *ACM Trans. Comput. Syst.*, 27(4):7:1–7:39, 2009.

[MMS99] Louise E Moser and Peter M Melliar-Smith. Byzantine-resistant total ordering algorithms. *Information and Computation*, 150(1):75–111, 1999.

[MNR19] Dahlia Malkhi, Kartik Nayak, and Ling Ren. Flexible byzantine fault tolerance. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1041–1053. ACM, 2019.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[NTT20] Joachim Neu, Ertem Nusret Tas, and David Tse. Ebb-and-flow protocols: A resolution of the availability-finality dilemma. *CoRR*, abs/2009.04987, 2020.

[YMR+19] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 347–356. ACM, 2019.

[ZRAP18] Vlad Zamfir, Nate Rush, Aditya Asgaonkar, and Georgios Piliouras. Introducing the minimal cbc casper family of consensus protocols. 2018.