Minimizing Mempool Dependency in PoW Mining on Blockchain: Scaling Blockchain Throughput via Asynchronous Pipeline Consensus

Gyu Chol Kim*

Faculty of Information Science and Technology, Kim Chaek University of Technology, Democratic People's Republic of Korea

Corresponding author: kcu201104@163.com , kgc841110@star-co.net.kp

ABSTRACT Traditional Proof-of-Work protocols face structural limits in transaction throughput, as scaling block capacity within a serial verification model introduces severe propagation delays and elevates orphan block rates. To address this classical constraint, this paper proposes an asynchronous pipeline consensus architecture that decouples mining initiation from full transaction verification. By embedding a 6-byte compressed UTXO identifier list within an agile Compact Block, nodes can initiate mining on subsequent blocks with reduced delay. This mechanism shifts data latency from a rigid physical bottleneck to a configurable declaration status, introducing a flexible validation framework. Network designers can thereby adjust the dissemination window—defined by the pipeline depth k and mining interval—to adapt to varying block sizes under the Nakamoto security model. Validated by game-theoretic analysis and a 1,000-node global emulation, the protocol demonstrates robust convergence under a work-weighted Difficulty Adjustment Algorithm patch, achieving approximately 66.7 TPS with 10MB blocks. Furthermore, we introduce a Dual-Coinbase mechanism that removes mandatory inter-block freezing, thereby unlocking the full potential of the pipeline parameter k and demonstrating applicability to alternative consensus architectures such as PoS, BFT, and DAG. This study establishes a backward-compatible upgrade path for high-performance decentralized ledgers by demonstrating that blockchain scalability can be optimized through the strategic configuration of its minimal consensus payload.

Keywords: Blockchain, Compact Block, Scalability, Pipeline Consensus, Proof-of-Work (PoW)

1. INTRODUCTION

The scalability of blockchain technology has been a primary focal point of global research in distributed ledger technology for over a decade [1-7]. While Bitcoin, proposed by Satoshi Nakamoto [8], established a foundation of decentralized trust through Proof-of-Work (PoW) and the Longest Chain Rule [9], it left behind the challenge of physical limits on transaction throughput (TPS). To address this, previous researchers have introduced innovative solutions from various perspectives. We build upon these contributions while proposing a new paradigm to overcome their inherent limitations.

The scalability of PoW blockchains has been a central research theme, with numerous scholars proposing diverse solutions over the past decade [1-3, 10, 11].

First, extensive research into Layer 2 and off-chain solutions has been conducted to increase throughput without directly modifying the base layer (Layer 1) [12]. State channel methods, such as the Lightning Network [13], decentralized traffic by processing microtransactions off-chain, while Rollup technologies sought to ensure data availability by compressing and recording transaction data on the main chain. These studies achieved the monumental feat of reducing the burden on the main chain while maintaining its security. However, they introduce secondary risks, including increased complexity in user experience and the intervention of centralized sequencers.

Layer 2 solutions like the Lightning Network and Rollups have significantly offloaded transaction pressure but introduce UX complexity and potential centralization risks [12]. Concurrently, research into data compression and network optimization (Light-net/Propagation) has followed, aiming to improve throughput by enhancing the propagation efficiency of the main chain. BIP 152 (Compact Blocks) [14] drastically reduced block propagation size by recycling transactions within the mempool, and research such as Graphene [15] achieved propagation efficiency near theoretical limits by introducing Invertible Bloom Lookup Tables (IBLT).

Furthermore, physical layer optimizations, such as the FIBRE block relay network [16], have significantly contributed to reducing propagation latency. These pioneering works have been instrumental in lowering the physical barriers of block propagation.

Innovative propagation technologies such as Compact Blocks and Graphene have minimized bandwidth usage but remain tethered to the serial verification model. Despite these remarkable advancements, a fundamental bottleneck remains: the "Serial Verification" structure. Previous compression technologies focused primarily on reducing block "size" but face inherent constraints in fully decoupling the "temporal dependency", which is the requirement that mining for block $n + 1$ must halt until the propagation and verification of block n are complete. In an era of large-scale blocks, this serial structure forces honest miners into "idle time," which inevitably increases the orphan block rate and erodes the network's security budget. Despite advancements in compression, the inherent serial bottleneck where mining $n + 1$ must wait for block n continues to restrict throughput and increase orphan rates. Building upon the technological foundations laid by previous researchers, this paper presents an architectural solution: the "asynchronization of the consensus structure," moving beyond mere propagation efficiency. The proposed scheme effectively mitigates the impact of propagation latency by absorbing it into the PoW computation time (Latency Hiding) through a post-confirmation mechanism where block $n + k$ validates block n.

Furthermore, we achieve a "logical isolation" that prevents the invalidation of a specific block from leading to the collapse of the entire chain. In the baseline design, this isolation is enforced by UTXO Freezing (Mempool Freezing). As an alternative that removes the mandatory freezing while preserving the same isolation guarantee, a dual-coinbase mechanism is proposed in Appendix B, thereby eliminating the cascading effect of block n's invalidation on subsequent blocks.

Our research proposes a structural shift towards asynchronous verification and logical isolation, hiding latency within the PoW process to break the serial bottleneck. Building upon prior work in block propagation and consensus optimization, this study proposes an asynchronous protocol designed to improve transaction throughput while maintaining decentralization properties. Without the need for complex layer separation or advanced cryptographic techniques, this model provides a meaningful contribution to PoW consensus, enabling higher transaction throughput while maintaining structural compatibility with Bitcoin’s core design.

The remainder of this paper is organized as follows. Section 2 reviews the existing literature on blockchain scalability and identifies the fundamental limitations of current consensus mechanisms regarding propagation latency. Section 3 introduces the proposed architecture, detailing the Extended Header, the Compact Block structure with 6-byte compressed IDs, the logical isolation mechanism, and the Context Specific Chain Selection Rules. Section 4 and 5 provide a security and performance analysis, evaluating the collision probability of compressed IDs and the improvements in network throughput and stability. Section 6 presents a discussion on the protocol's extensibility, including block interval reduction and potential economic incentive models for honest declarations. Section 7 concludes the paper by summarizing the key contributions and suggesting future research directions.

2. RELATED WORK

This section categorizes and reviews the previous research conducted over the past decade to resolve blockchain scalability issues into three major categories: data propagation, consensus structures, and off-chain solutions.

2.1. Propagation and Bandwidth Optimization Research

Pioneering research has been conducted to overcome bandwidth constraints and propagation delays, which serve as the primary barriers to blockchain scalability.

BIP 152 (Compact Blocks) [14]: This protocol significantly enhanced block propagation efficiency by transmitting 6-byte short IDs instead of full transaction data, reducing the propagation size by over 90%. While it has become a standard, it possesses a critical limitation: if the mempool states between nodes are inconsistent, additional data requests (getblocktxn) are triggered. This results in unpredictable round-trip latencies, exacerbating propagation delays.

Graphene and Xthin [15]: These approaches utilized Bloom Filters and IBLT to compress propagation data toward its theoretical lower bound by leveraging mempool knowledge. However, the asynchronous nature of mempools often leads to decoding failures and subsequent synchronization overhead. More importantly, these methods remain bound to the inherent serial structure where mining must halt whenever a propagation delay occurs.

Falcon and FIBRE [16]: These solutions attempted to reduce block propagation latency to tens of milliseconds by utilizing fiber-optic networks and relay servers at the physical layer. While they contributed significantly to hardware-level optimization, they entail high costs and centralization risks, making them difficult to apply universally across a global, decentralized node environment.

2.2. Consensus & Parallelization

Innovative consensus algorithms have been proposed to enhance processing speed by refining the conventional serial block production structure.

Bitcoin-NG [10]: This protocol decoupled blocks into "Key Blocks" for leader election and "Microblocks" for transaction records. By allowing a leader node to continuously produce microblocks until the next key block is mined, it successfully increased throughput. While this mitigated the dependency between propagation delay and mining, it introduced new challenges, such as the risk of Denial-of-Service (DoS) attacks on leader nodes and increased verification complexity.

GHOST (Greedy Heaviest Observed Subtree) [11]: This algorithm sought to prevent security degradation caused by propagation delays by including "Uncle Blocks" (orphaned blocks) in the cumulative weight of the chain. Although adopted by networks like Ethereum to enhance security, its impact was limited to accommodating time-lagged blocks to maintain security under high orphan rates rather than maximizing true parallel data availability.

DAG (Directed Acyclic Graph) [17]: Nonlinear structures attempted in projects like IOTA and Fantom sought theoretical infinite parallelism by allowing blocks to reference each other in a mesh-like structure. However, this approach significantly complicates the determination of deterministic transaction order (finality) and double spend validation, often compromising the blockchain's core value of simple and transparent consensus.

2.3. Layer 2 and Data Availability Solutions

A significant body of research focuses on achieving scalability by building upper layers rather than altering the fundamental design of the main ledger.

Payment Channels (Lightning Network) [13]: This approach enables near infinite off-chain transactions by opening dedicated channels between parties, with only the final settlement recorded on-chain. While it has revolutionized micropayment efficiency, it faces operational constraints such as liquidity locking and complex routing requirements.

Sharding [18]: By partitioning the network into multiple "shards," this method allows for distributed transaction processing. Although it significantly increases throughput, it introduces fragmentation of the security model and adds immense complexity to cross-shard communication and defense against single shard attacks.

Rollups (Optimistic & ZK) [19]: Rollups perform computations off-chain and submit only summarized data or zero knowledge proofs to the main chain. While these are currently widely adopted scaling strategies for modern blockchains, they remain inherently limited by the performance of the main chain, as the cost of ensuring Data Availability (DA) is still tethered to the base layer's capacity.

3. PROPOSED SYSTEM DESIGN

This section details the technical specifications of the proposed asynchronous pipeline consensus and latency-hiding framework. To provide a concrete and intuitive understanding of the architectural mechanics, the system's operations are illustrated through a representative deployment example featuring a 10MB Full Block size and a Bitcoin-standard 600-second mining interval.

Crucially, this 10MB and 600-second configuration serves as an illustrative exemplar rather than a rigid physical constraint of the protocol. The core contribution of this framework is the generalized, consensus-agnostic validation layer. To highlight this structural shift, the technical design is presented in continuous juxtaposition with the legacy Bitcoin model. While traditional Bitcoin remains tethered to a serial verification loop—where block $n + 1$ mining is physically bottlenecked by block n's propagation—our system breaks this temporal dependency. By decoupling the lightweight Compact Block consensus from the background Full Block availability, the pipeline parameter k, the propagation gate i, and the block specifications can be jointly optimized. Therefore, the 10MB/600s configuration utilized in this section is merely a baseline to demonstrate how the protocol hides extreme physical transmission delays under the Nakamoto security model, thereby establishing a scalable paradigm applicable to diverse ledger specifications.

3.1. Data Structure Design

The proposed protocol utilizes three distinct types of data structures, which differ from the standard Bitcoin protocol, to enhance efficient block propagation and verification. These structures form the core components of the protocol's operation.

3.1.1. Block Header

The Block Header is the most critical fixed-length data structure in the proposed protocol and serves as the target for the PoW computation. Furthermore, it acts as a logical linkage between the Compact Block (CB) and the Full Block (FB). The block header of the proposed system strictly inherits the standard 80 bytes structure of Bitcoin while appending 33 bytes for asynchronous finality and tamper proofing, resulting in a total of 113 bytes. These additional fields allow the miner to certify the validity of the $(n-k)$-th block and enable the receiver to immediately detect any tampering of the Compact Block.

Table 1: Detailed Description of Block Header Fields

Field Name	Size (Bytes)	Description	Remarks
Version	4	Block version number	Inherited from Bitcoin
Prev_Block_Hash	32	Hash of the previous $(n-1)$-th block header	Inherited from Bitcoin
Merkle_Root	32	Merkle tree root of all transactions (FB)	Inherited from Bitcoin
Timestamp	4	Block generation time (UNIX Epoch)	Inherited from Bitcoin
Difficulty_Target	4	The difficulty target value for the current n-th block.	Inherited from Bitcoin
Nonce	4	Arbitrary value for PoW computation	Inherited from Bitcoin
$n_k_declaration$	1	Declaration of $(n-k)$-th block validity (0: False, 1: True)	Key Mechanism
Comp_UTXO_Root	32	Merkle root of the Compressed UTXO ID List	Key Mechanism

Key Features and Role:

PoW Target: The entire 113bytes data, including all the above fields (especially the nonce), is used as input for the SHA-256 algorithm. The resulting hash value must be less than the difficulty target.

Security and Integrity: The Merkle_Root and Comp_UTXO_Root provide strong cryptographic evidence of the Full Block and Compact Block contents, respectively, thereby preventing data tampering.

3.1.2. Compact Block

The Compact Block is assigned the highest priority for network propagation, enabling nodes to decide on immediate mining entry and transaction freezing even before the Full Block is received. The 6-byte UTXO compressed ID list ensures that even for large scale blocks, the transmission volume remains in the range of tens of kilobytes (KB), supporting large scale scalability.

Table 2: Detailed Description of Compact Block Fields

Field Name	Size (Bytes)	Description	Role
BlockHeader	113	As defined in Section 3.1.1	PoW verification and $n_k_declaration$ check

Input_UTXO_Count	Variable (1-9)	The total number of UTXOs consumed as inputs by the transactions within the block	Defines the length of the Compressed ID list
UTXO_Comp_IDs	Variable	6-byte compressed ID list of input UTXOs	Mempool Freezing filter

Key Features and Role:

Immediate Mining Trigger: Enables nodes to commence mining the next block immediately upon receiving the Compact Block, effectively eliminating idle time caused by Full Block propagation.

Logical Isolation: Facilitates instantaneous Mempool Freezing by using the Input_UTXO_Count and compressed ID list, ensuring logical decoupling between adjacent blocks without full data.

3.1.3. Full Block

The Full Block contains the complete data of all transactions to be permanently recorded on the blockchain ledger. It propagates relatively slowly in the background after the compressed block has been disseminated, and nodes use it to perform 'delayed verification'. The Full Block serves as the definitive data entity of the chain and is the subject of verification for post confirmation at the $n + k$ stage. The Full Block proves its adherence to the logical isolation rules by demonstrating that it does not utilize any UTXOs generated after the $n - k$ block, thereby ensuring the stability of the asynchronous structure.

Table 3: Detailed Description of Full Block Fields

Field Name	Size (Bytes)	Description	Role
Block Header	113	The block header structure defined in **Table 1**. It shares the same header as the Compact Block.	Cryptographic binding of data and header
Transaction Count	Variable (1-9)	The total number of transactions	Ledger recording and final validity check
Transaction Data	Variable	All actual transaction data within the block	

Key Features and Role:

Basis for Full Verification: Upon receiving this block, a node performs final validation of all transactions, including signatures and UTXO status.

Data Storage: Once verified, the Full Block is stored in the local database, constituting the blockchain ledger.

3.2. Asynchronous Propagation & Mining Process

This protocol independently decouples block propagation speed from mining continuity, ensuring that the entire network's effective hash power operates without idle time.

3.2.1. Dual Propagation & Immediate Mining

Simultaneous Broadcast: A successful miner broadcasts both the $n - th$ Compact Block and Full Block simultaneously. Due to its minimal size, the Compact Block propagates with extreme agility across the network before the Full Block transmission gains momentum.

Preemptive Verification & Mining Entry: Neighboring nodes perform the "Compact Block Validation" (as detailed in Section 3.2.3) within milliseconds of receiving the Compact Block. If valid, the nodes immediately commence mining the $(n + 1) - th$ block, bypassing the need to wait for the arrival of the large $n - th$ Full Block.

Latency Hiding: The physical propagation delay of the Full Block is effectively hidden within the PoW computation interval of the subsequent blocks. This architecture ensures that even with large-scale blocks, the occurrence of propagation-triggered orphan blocks is substantially reduced toward baseline minimums. The figure below illustrates the difference between the conventional serial verification based blockchain and the asynchronous pipeline consensus architecture proposed in this paper.

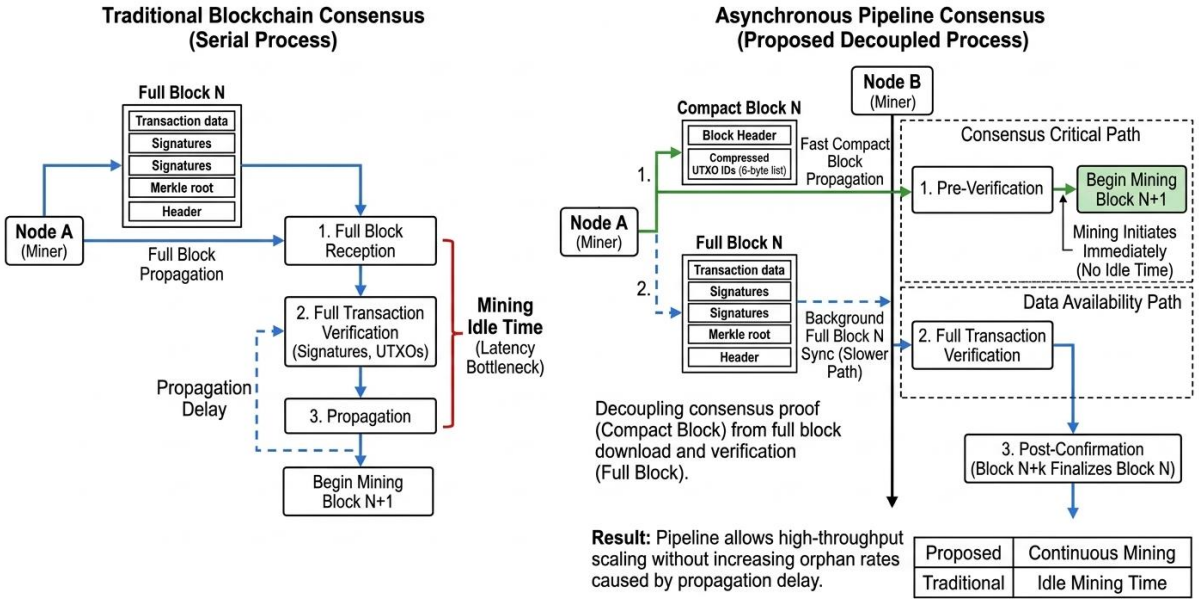


Figure 1. Serial Full-Block Verification Baseline versus the Proposed Asynchronous Pipeline Consensus.

3.2.2. Accelerated Convergence via Coupling

A. Coupling Mechanism and Propagation Rules

Universal Propagation: If the $n-th$ Full Block is valid, it is propagated throughout the entire network until all nodes receive it.

Coupling Condition: If the $n-th$ Full Block encounters an $(n+k)-th$ Compact Block with $n_k_declaration = True$ at an intermediate node, they are coupled into a single packet and propagated together. (In most cases, the $n-th$ Full Block will have already saturated the network before the $(n+k)-th$ Compact Block is mined).

B. Impact on Specific Node States

This coupled propagation specifically targets and influences nodes that are currently mining or have already mined the $(n+k)-th$ block with a “False” declaration due to data latency.

Nodes in Progress $n+k$ Mining with “False”: Nodes that were mining the $n+k$ block while assuming the $n-th$ Full Block was missing ($n_k_declaration$ = False) will immediately switch their declaration to “True” upon receiving the $n-th$ Full Block. They then restart or pivot their mining process toward the $n+k$ block with the updated “True” status. Furthermore, if they receive a coupled packet (containing both the $n-th$ Full Block and the $n+k$ Compact Block), they immediately transition to the "True chain" and commence mining the $n+k+1$ block.

Nodes with Completed Blocks (Mining $n+k+j$ with “False”): Nodes that have already succeeded in mining the $n+k$ block with $n_k_declaration$=False and are currently working on subsequent blocks ($n+k+j$, where $j>0$) do not perform an immediate chain switch upon receiving the $n-th$ Full Block or the coupled packet. Instead, they record the arrival of the data and detect that a fork has occurred. These nodes will only undergo a reorganization (Reorg) if the “True chain” eventually proves to be the longest chain according to the consensus rules.

3.2.3. Verification Sequence

This section describes the strict step-by-step verification procedures used by nodes to determine the legitimacy of received data, ensuring integrity in high throughput environments. As with the Bitcoin protocol, any data unit in our scheme including

transactions, Compact Blocks, and Full Blocks is ignored if it fails verification. Such invalid data is neither propagated further through the network nor retained, resulting in its immediate elimination from the system.

A. Header Validation

First, the node verifies that the block header's PoW meets the current network difficulty target. It also checks that the $n_k_declaration$ field complies with the binary specification (0 or 1).

B. Compact Block Validation

A preliminary verification phase performed on the Compact Block alone, prior to Full Block arrival (Figure 2).

Freeze Duplicate Check: The node checks if the compressed UTXO IDs in the Compact Block overlap with elements already in the local node's Frozen Set (See 3.3.1). If a duplicate is found, it is flagged as a double spend attempt and the Compact Block is rejected immediately.

Merkle Root Comparison: The node constructs a Merkle tree from the received compressed ID list and verifies that the resulting root matches the Comp_UTXO_Root in the header.

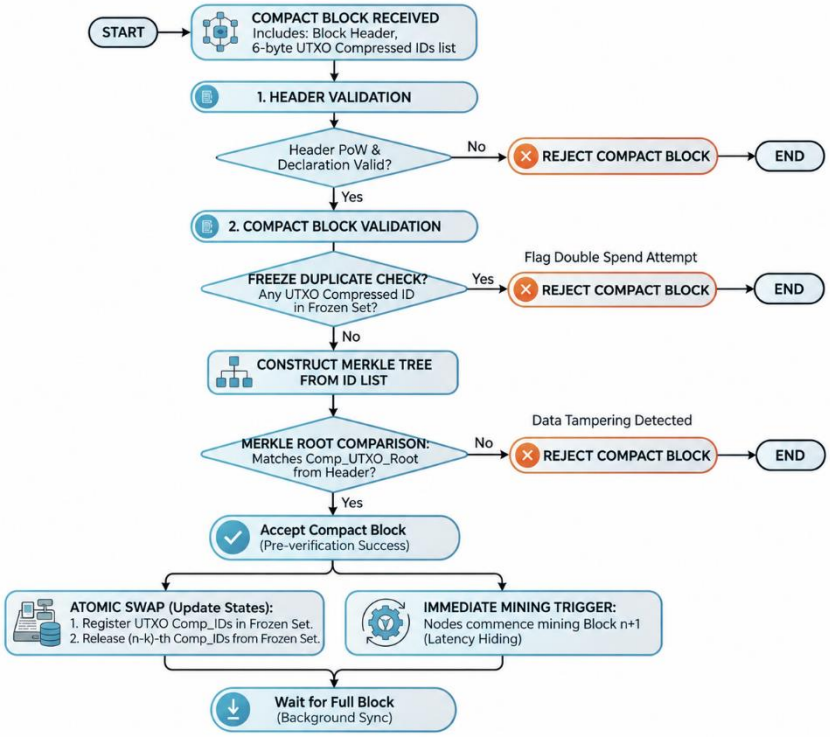


Figure 2. The Validation and Processing Procedure of a Compact Block.

C. Full Block Validation

A deep verification phase for the definitive data entity.

Transaction Validity: All transaction signatures and script integrity within the block are verified.

Compressed List Consistency: The 6-byte compressed IDs derived from the Full Block's actual transaction inputs must be a 100% match with the Compact Block's list. Any discrepancy is treated as data tampering, and the block is invalidated.

Isolation Rule Check: The node verifies whether the transactions in the $n-th$ Full Block consume UTXOs generated during the logical isolation period ($n-k$ to $n-1$). The $n-th$ Full Block is strictly prohibited from consuming any UTXOs generated during the logical isolation period (from block $n-k$ to $n-1$) that have not yet been finalized as “True”. Any block violating this rule is discarded immediately (See 3.3.3 for more details).

3.2.4. Strategic Propagation Delay for Bandwidth Optimization

To further enhance network efficiency and eliminate redundant data transmission during transient forks, this protocol implements a "Strategic Propagation Delay" for Full Blocks.

A. i-Confirmation Gated Propagation

The propagation of the $n-th$ Full Block is strictly gated by the cumulative PoW of its corresponding Compact Block chain. A miner or a node shall only commence the universal broadcast of the $n-th$ Full Block after the $n-th$ Compact Block has been included in the longest chain and has achieved i subsequent confirmations (i.e., at the height $n+i$). For general network stability, $i=2$ is established as the default threshold.

B. Impact on Fork Resolution

Suppression of Orphan Data: During a brief fork at height n, multiple competing Compact Blocks may exist. By withholding the heavy Full Blocks until the Compact Block level conflict is resolved (usually within 1-2 blocks), the network mitigates the bandwidth contention caused by the simultaneous propagation of multiple, conflicting heavy Full Blocks.

Winner Takes All Efficiency: Once the $(n+i)-th$ Compact Block is mined, it signals a definitive PoW consensus on which n-th block is the legitimate ancestor. Only the Full Block belonging to the canonical chain is propagated through the network. This ensures that 100% of the network's bandwidth is dedicated to valid, persistent data.

Deterministic Arrival Window: Since the $n-th$ Full Block starts its journey at the $n+i$ mark, it still possesses a $r(=k-i)$ block window (approximately 40 minutes in a 10-minute interval system when $i=2$ and $k=6$) to reach all nodes before the final $n+k$ evaluation. This provides a robust safety margin that reconciles the physical limits of data transmission with the logical requirements of consensus. Even if the $(n+k)-th$ Full Block is not received by the time $n+2k$ is reached, honest nodes will declare $n_k_declaration$ = False in the $(n+2k)-th$ block based on their observations. Consequently, the $(n+k)-th$ block is deterministically rejected (switched off), allowing the pipeline to proceed to the next block without interruption.

The Nature of the Gated Propagation Mechanism: The i-confirmation gated propagation mechanism is not strictly required for the correctness of the consensus protocol itself. Rather, it serves as a network-level stabilization mechanism that reduces propagation contention and mitigates data flooding risks associated with large Full Blocks.

3.3. UTXO Freezing and Logical Isolation

This protocol applies dynamic freezing control at the mempool level and isolation principles to asynchronously manage data dependencies between blocks.

3.3.1. Definition of the Frozen Set and Atomic Update Logic

This section defines the atomic mechanism that simultaneously handles Asset Isolation (Freezing) for future mining and State Finalization for past blocks, triggered by the arrival of a Compact Block.

A. Definition of the Frozen Set

The Frozen Set is defined as a collection of compressed UTXO IDs that are restricted from being used as inputs during the k-block isolation period. It functions as a dynamic input filter to ensure transaction consistency.

B. Atomic Swap of IDs (Simultaneous Register & Release)

Upon receiving the $n-th$ Compact Block, the node executes the following two operations as a single atomic transaction to maintain optimal mempool efficiency:

Immediate Release (Release $n-k$): The node removes the UTXO_Comp_IDs associated with the $(n-k)-th$ block from the Frozen Set. Since the $n-th$ header already contains the fate of the $(n-k)-th$ block via the $n_k_declaration$, there is no technical necessity to maintain these IDs in the isolation list.

New Registration (Register n): Concurrently, the UTXO_Comp_IDs from the $n-th$ Compact Block are immediately added to the Frozen Set. This prevents the $(n+1)-th$ block from including any transactions that conflict with the $n-th$ block.

C. Candidate Block Construction Principles

When constructing the $(n+1)-th$ candidate block, mining nodes apply the following filtering rule to the mempool.

Rule: "Any transaction that contains a UTXO_Comp_ID belonging to the Frozen Set as one of its inputs must be excluded from Selection (Figure 3)."

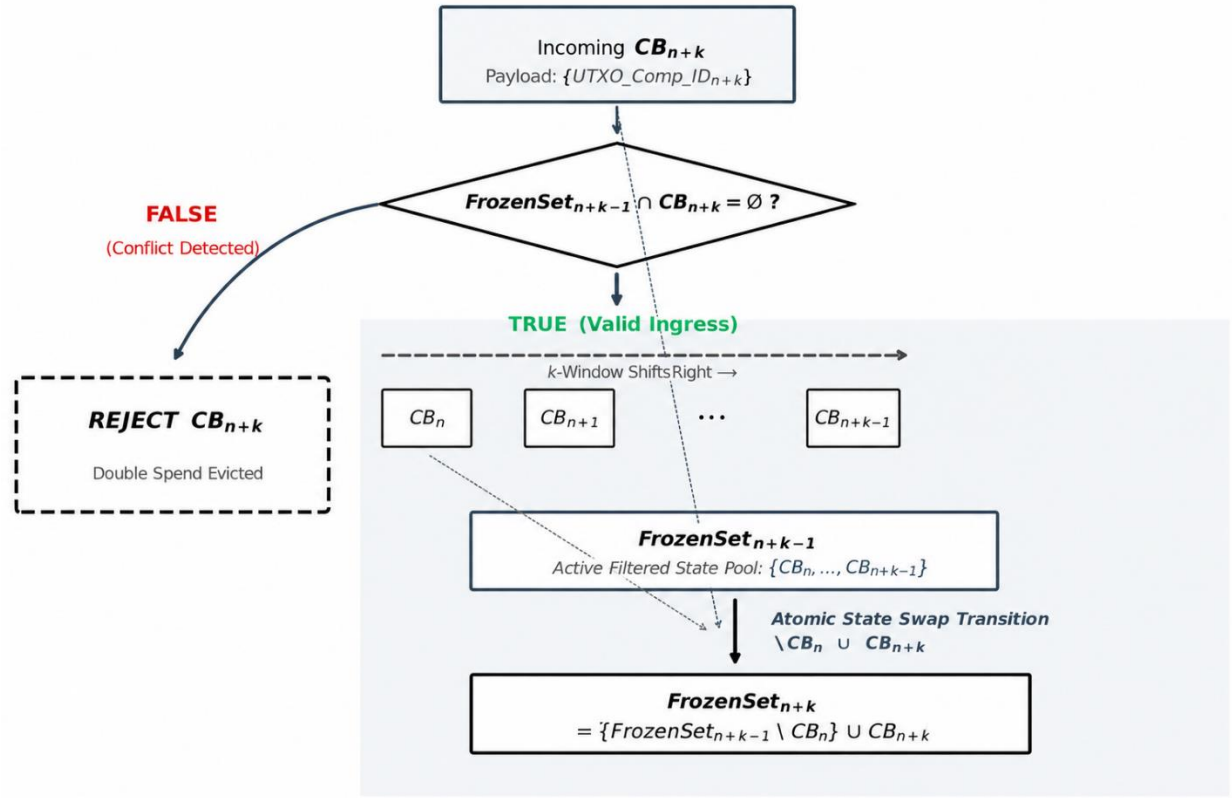


Figure 3. Frozen Set and Atomic Update Logic.

D. Acceptance of Over Freezing

Due to the nature of 6-byte (48-bit) compressed IDs, accidental hash collisions (as explained by the Birthday Paradox) may cause unrelated transactions to be frozen. However, this protocol adheres to the strict security principle of "allowing false positives (over freezing) while prohibiting false negatives (missed freezing)". Since an accidentally frozen transaction is simply released after k blocks to be included in a later block, this conservative freezing is accepted as a necessary trade-off to enhance overall system integrity and stability.

3.3.2. Consistency and Release of Freeze

Concurrent with the release of IDs from the Frozen Set, the node finalizes the global ledger (UTXO Set) based on the $n_k_declaration$ value found in the $n-th$ Compact Block.

A. Case 1: $n_k_declaration$=True (Finalization)

Ledger Update: The actual UTXOs associated with the $(n-k)-th$ block are permanently marked as "Spent" in the ledger.

Mempool Cleanup: Transactions that were successfully included and finalized are purged from the Mempool to optimize memory resources.

B. Case 2: $n_k_declaration$=False (Invalidation & Recovery)

Asset Restoration: As the $(n-k)-th$ block is deemed invalid, the transactions associated with it are restored to a "spendable" status within the Mempool.

Isolation Integrity: Due to the logical isolation, blocks from $(n-k+1)$ to $(n-1)$ remain unaffected by this invalidation. The restored assets are eligible for inclusion starting from the $(n+1)-th$ candidate block.

PoW Preservation: Although the Full Block of an invalidated block is discarded and its transactions are restored to the mempool, the corresponding header remains embedded in the canonical chain, and its proof-of-work continues to contribute to the cumulative work that secures the chain. This property mirrors the treatment of uncle blocks in GHOST-style protocols [11], in which orphaned blocks are excluded from the canonical transaction history yet remain part of the cumulative weight.

3.3.3. Architectural Isolation Rules for Logical Decoupling

This section defines the core isolation rules necessary to physically break the data dependencies and logically decouple blocks, ensuring integrity within the asynchronous mining environment.

No Chaining Rule (Prohibition of New Asset Use): Any new UTXOs generated within the $n-th$ block (including change outputs) are in an unconfirmed state until finalized at block $n+k$. Therefore, these UTXOs absolutely cannot be used as inputs in any transactions within blocks $n+1$ through $n+k$. This extends Bitcoin's coinbase maturity concept (100-block wait) to general transactions for a duration of k blocks.

Independent Validity Guarantee: Adherence to this rule ensures that even if the $n-th$ block is invalidated ($n_k_declaration$=False), all subsequent blocks (from $n+1$ onwards) maintain their independent validity. They are guaranteed to have been created only using UTXOs already finalized before the $n-k$ point. This prevents cascading invalidation and maximizes the system's resilience.

The consensus layer enforces a delayed deep verification loop: when the mining frontier reaches Block $n+2k$, individual nodes execute a deterministic lookup audit on Full Block B_{n+k} using the aggregated history of all preceding blocks to detect and invalidate any chaining rule violations, thereby maintaining structural decentralization liveness without cascading invalidation vulnerabilities (Figure 4).

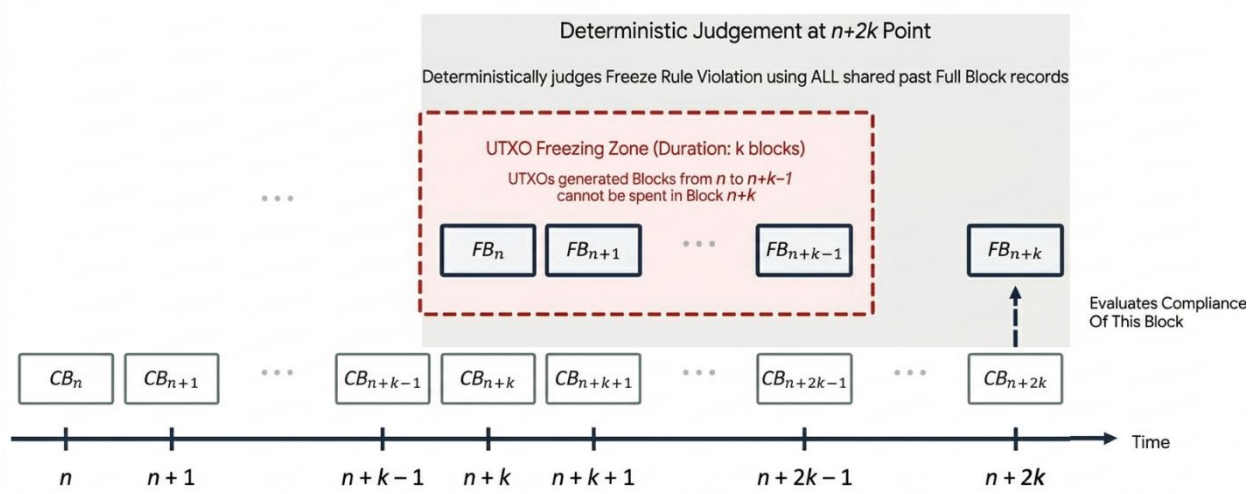


Figure 4. Pipeline Timeline of Architectural Isolation Rules.

3.4. Fork Convergence Algorithm

To rapidly integrate temporary branches in an asynchronous environment, this protocol inherits Bitcoin's Longest Chain Rule but integrates a unique convergence mechanism combined with "Data Availability Verification."

3.4.1. Pipelined Consensus Mechanism

This section formally defines the pipelined consensus mechanism, which is described in Section 3.1, 3.2 and 3.3. This mechanism overcomes the sequential constraints of legacy systems by decoupling mining from finalization through a k-stage temporal phase shift.

A. Notations and Definitions

B_n, CB_n: $n-th$ Full Block and Compact Block.

S_n: Ledger state finalized up to block n. In Bitcoin, this refers to the finalized UTXO set.

TX_n: Mempool state.

C_n: Filter state (Frozen Set) updated via CB_n.

$NewTX_n$: Set of newly incoming transactions.

$CB_n.n_k_declaration$: Validity declaration for block $(n-k)$ in CB_n header.

$t_{start}(B_n)$, $t_{end}(B_n)$: Start and end times of B_n mining.

$Pow(B_n)$: Pure computation time for mining $t_{end}(B_n) - t_{start}(B_n)$.

$T(B_n), T(CB_n)$: Propagation times for B_n and CB_n to reach >90% of nodes.

F, G, H: Functions for state transition, mempool update, and filter update.

B. Bitcoin's Sequential Consensus Model ($k = 0$)

In Bitcoin, consensus proceeds atomically and sequentially on a single timeline. The transition is as follows:

Transitions:

1. **Mining:** $B_n = BlockGenerate(S_{n-1}, TX_{n-1})$
   A miner generates block B_n by assembling transactions from the current mempool TX_{n-1} atop the finalized historical ledger state S_{n-1}. This process cannot commence until the prior state S_{n-1} has been fully received and validated.

2. **Finalization:** $S_n = F(S_{n-1}, B_n)$
   Upon reception of B_n (provided it is valid), a node instantly updates its local ledger state from S_{n-1} to S_n. Crucially, the mining of the subsequent block cannot begin until this ledger update is complete.

3. **Mempool Update:** $TX_n = G(TX_{n-1}, B_n, NewTX_n)$
   Transactions included in B_n are purged from the mempool, and newly arrived transactions $NewTX_n$ are added to the waiting pool.

Temporal Constraint: $t_{start}(B_{n+1}) = t_{end}(B_n) + T(B_n)$

The start of mining for B_{n+1} is directly dependent on the full propagation of B_n, making the system highly sensitive to propagation latency. A fundamental throughput bottleneck in the Nakamoto consensus model stems from the sequential dependency of its mining initiation point. Consequently, the $(n+1)-th$ block is forced to wait for the duration of $T(B_n)$, the time required for the $n-th$ block to propagate fully across the network. This bottleneck is the direct cause of the sharp decline in total network throughput as block sizes increase and $T(B_n)$ lengthens. While Bitcoin's Compact Block Relay (BIP 152) attempts to mitigate this by reducing $T(B_n)$, it remains constrained by the structural limitations inherent in the sequential architecture.

C. Proposed Pipelined Consensus Model ($k = 6$)

The proposed model separates mining initiation from ledger finalization using a k-stage pipeline.

Transitions:

1. **Mining:** $(B_n, CB_n) = BlockGenerate(S_{n-k-1}, TX_{n-1}, C_{n-1}, CB_{n-1})$
   The $n-th$ block is mined based on the finalized state S_{n-k-1} from $(n-k-1)-th$ stages prior, while simultaneously integrating the latest mempool TX_{n-1} and the dynamic filter C_{n-1} to ensure transaction consistency.

2. **Filter Update:** $C_n = H(C_{n-1}, CB_n)$
   Upon reception, the Compact Block CB_n immediately updates the Frozen Set state C_n, which serves as the filter for the subsequent block $n+1$. This process is completed within the minimal duration of $T(CB_n)$.

3. **Finalization:** $S_{n-k} = F(S_{n-k-1}, B_{n-k}, CB_n.n_k_declaration)$
   At the moment the $n-th$ block emerges, the ledger state transitions from S_{n-k-1} to S_{n-k} according to the validity declaration $n_k_declaration$ assigned to the $(n-k)-th$ block.

4. **Mempool Update:** $TX_n = G(TX_{n-1}, B_{n-k}, CB_n.n_k_declaration, NewTX_n)$
   Based on the data of the finalized block B_{n-k}, the mempool is updated by purging($CB_n.n_k_declaration$=True) or restoring($CB_n.n_k_declaration$=False) consumed transactions and accepting newly incoming transactions $NewTX_n$.

A conventional pipeline operation follows the timing rule:

$$t_{start}(B_{n+1}) = max\{t_{end}(B_n) + T(CB_n), t_{end}(B_{n-k+1}) + T(B_{n-k+1})\}$$

When considering the i-confirmation-gated propagation, the aforementioned equation is modified as follows:

$$t_{start}(B_{n+1}) = max\{t_{end}(B_n) + T(CB_n), t_{end}(B_{n-k+1+i}) + T(B_{n-k+1})\}$$

In this model, if the right-hand term representing the propagation of the heavy Full Block $T(B)$ exceeds the left-hand term, a pipeline stall occurs, forcing the system to wait and wasting network resources.

In the proposed system, the arrival status of the Full Block is decoupled from the mining initiation trigger. Consequently, the start time is always pegged to the left-hand term:

$$t_{start}(B_{n+1}) = t_{end}(B_n) + T(CB_n)$$

The node does not treat the reception of B_{n-k+1} as a prerequisite for mining. Instead, it simply selects the declaration value "True" if the data has arrived, or "False" if it has not and continues the mining process without interruption.

t_{start} is determined exclusively by the Compact Block CB_n, which has a minimal and consistent propagation delay.

Any delay in $T(B_{n-k+1})$ affects only the 1-bit declaration value $n_k_declaration$. In essence, data latency becomes a matter of "what to declare" rather than "when to start."

If the propagation delay $T(B_{n-k+1})$ exceeds the physical threshold of the k-window, the system opts for asynchronous forking instead of a pipeline stall. This prioritizes the Liveness of the network, ensuring that consensus momentum is never sacrificed for data synchronicity.

As a result, the proposed mechanism ensures that the block production continuity of the network is maintained even when data moves slower than the mining speed. Since $T(CB) \ll T(B)$ on average, t_{start} is effectively pegged to the propagation speed of the Compact Block.

Note. If the block size, mining interval, k and i are configured appropriately (e.g., block size of 10MB, mining interval of 600s, $k = 6$ and $i = 2$), $t_{end}(B_n) + T(CB_n) > t_{end}(B_{n-k+1+i}) + T(B_{n-k+1})$ holds with high probability(See Section 5.2). Even if a fork occurs due to the delay of B_{n-k+1}, the system operates on the premise that any forks will be resolved through the longest-chain rule and Proof of Work competition within the sufficient temporal buffer provided by k. The fork convergence algorithm is described below.

3.4.2. Context-Specific Chain Selection Rules

When a node discovers a longer chain (B) of length b_len than its current one (A) of length a_len, it follows a sophisticated algorithm to prevent ghost chain attacks by integrating data availability checks into the longest chain rule. Assuming that Group A and Group B chains are consistent up to the $n - th$ block (the fork point), the relative progress after the fork is represented as $a_z = a_len - n$ for Group A and $b_z = b_len - n$ for Group B. Let $m = b_len - a_len = b_z - a_z$.

A. Threshold Convergence ($m = 1$ and $a_z = 0$)

At the moment a fork starts, the node performs an "objective verification based on subjective observation." It compares the $n_k_declaration$ in the $b_len - th$ Compact Block of chain B with its local data status. The protocol strictly defines the verification timing to ensure consistency between subjective observation and objective consensus. At the exact moment a node receives the $b_len - th$ Compact Block, it performs the following steps:

Local Status Evaluation: The node checks the status of the $b_len - th$ Compact Block of chain B with its local data status.

If the $(b_len - k) - th$ Full Block is present and valid, the local status is set to "True".
Otherwise, the local status is set to "False".

Decision Logic: The node compares this local status with the $n_k_declaration$ field in the $b_len - th$ Compact Block header.

If the local status matches the declaration, the $b_len - th$ Compact Block is accepted.
Otherwise, the block is rejected immediately.

The switch only occurs if the declaration matches the actual data availability. This prevents confusion caused by "False" declarations without data.

B. Deterministic Convergence ($m = 1, a_z > 0$ or $m > 1$)

If a clear length difference exists, the longest chain rule is prioritized under conditional acceptance.

False-declared blocks: The node accepts the B chain's “False” evaluation to maintain consistency, even if it possesses the data.

True-declared blocks: The node must acquire and verify all Full Block data for blocks declared as “True” before finalizing the switch. No matter how long a chain is, it is never accepted without the underlying data.

Nodes only switch to a longer chain after verifying all data for blocks declared as “True”, preventing ghost chain attacks. In the event of a prolonged network partition, the fork resolution process via the longest chain rule necessitates significant overhead, as nodes must download and verify a vast number of accumulated blocks. This characteristic is inherent to any Nakamoto-consensus-based protocol, including Bitcoin.

3.4.3. Resolving Information Asymmetry via Coupling Mechanism

Bottleneck Elimination and Sync Efficiency: The Coupling Mechanism (as detailed in Section 3.2.2.) significantly mitigates the standby time required for "data re-requests and downloads" during a fork switch. Among honest nodes, data and consensus proof are organically shared via Coupling Mechanism. Consequently, upon receiving a new block, consensus convergence occurs through Compact Block verification alone, eliminating additional data request cycles. This terminates forks faster than the original Bitcoin protocol.

Resilience against Block Withholding: Even if an attacker attempts to deceive the network with a "longer chain" by leveraging high hash power while withholding the actual data (Full Block), the "Mandatory Data Verification" rule ensures that nodes without the data will not accept the attacking chain. As a result, the malicious chain is rejected by the majority of honest nodes and becomes isolated due to severe propagation bottlenecks.

This algorithm provides incentives for honest nodes through "fast propagation" while imposing physical barriers propagation delays and resource exhaustion on malicious actors. This serves as an economic and technical deterrent, forcing attackers to incur significantly higher costs to compete with honest participants.

3.5. Incentive Realignment and Security Stabilization via Punitive DAA(Difficulty Adjustment Algorithm) Patches

While the proposed architecture demonstrates high structural efficiency under a baseline assumption of universal honesty, its economic incentive model can be further enhanced to address potential strategic deviations. Specifically, if a miner broadcasts an ungrounded False declaration, the current scheme lacks an immediate, tangible deterrent. Under such conditions, the ledger remains stable as long as the majority remains strict to honest rules, but a deviating node faces no instant, direct physical friction. The baseline configuration relies primarily on ex-post incentive alignments, which can be modeled as a Weak Nash Equilibrium.

To systematically reinforce the protocol's security framework and establish a Strict Nash Equilibrium, the system introduces an inline target modulation rule to disincentivize groundless False declarations. This refinement shifts the paradigm from a delayed warning mechanism to an active, real-time computational deterrence via the following two interconnected execution rules governed by a penalty coefficient $\eta(\eta > 1)$:

3.5.1. Real-Time Physical Penalty via Difficulty Target Scaling

To minimize the strategic leeway for ungrounded False declarations, defection should incur an immediate computational cost. The deterrence against inaccurate declarations scales linearly with the penalty coefficient η.

We introduce an inline target modulation rule where any block carrying a False declaration is subjected to a difficulty penalty scaled by this coefficient η:

$$T_{penalty} = T_{normal}/\eta$$

where T_{normal} denotes the baseline difficulty target dynamically adjusted by the network to maintain the standard block generation interval, and $T_{penalty}$ represents the temporarily restricted, harder difficulty target applied exclusively to the block space validating the False declaration.

By scaling the mining puzzle difficulty by a factor of η, the effective hash rate of the defecting faction is attenuated at the moment of computation. Letting H_{adv} and H_{hon} represent the raw hashing power of the adversary and the honest nodes respectively ($H_{adv} + H_{hon} = 1$), the adversary’s effective network dominance Φ_{adv} within the False block space is formulated as:

$$\Phi_{adv} = \frac{\frac{H_{adv}}{\eta}}{\frac{H_{adv}}{\eta} + H_{hon}} = \frac{H_{adv}}{H_{adv} + \eta \cdot H_{hon}} < \frac{H_{adv}}{H_{adv} + H_{hon}} = H_{adv}$$

This adjustment ensures that deviating miners face direct financial friction through increased energy consumption and hardware depreciation the exact second they choose to defect.

3.5.2. Work-Weighted DAA Patch

Because the punitive target introduces heterogeneous work weights into the consensus epoch, a standard nominal block-counter would miscalculate the aggregate network hash rate, potentially leading to an unintended difficulty target variance. To neutralize this structural loophole and guarantee macro-economic stability, the DAA should be patched with a work-weighted denominator variant triggered strictly upon the completion of each standard epoch interval defined by:

$$N_{true} + N_{false} = 2016$$

When this target block count is reached, the next difficulty target is adjusted according to the following normalized formulation:

$$T^{+}_{normal} = T_{normal} \cdot \frac{t}{t_0} \cdot \frac{2016}{N_{true} + N_{false} \cdot \eta}$$

where T^{+}_{normal} is the target for the next round, while t_0 and t denote the desired and actual time consumed to generate 2016 blocks, respectively.

By normalizing the heavier False blocks into their effective work equivalents, this formula mathematically isolates the latency variance from the true network capacity. Furthermore, this creates a dynamic macro-economic constraint: a high frequency of False declarations expands the denominator, which compresses the target value and inherently triggers a network-wide difficulty spike in the subsequent epoch. The defecting miners are thus bound by a compounding economic constraint, suffering both immediate hash attenuation in the current epoch and diminished mining yields in the next.

3.5.3. Upper Bound Calibration of the Penalty Parameter η

While a larger η maximizes the deterrence against groundless False declarations, its upper bound must be strictly calibrated due to two critical boundary conditions that restrict hyper-punitive configurations:

First, an excessively large η induces an adversarial configuration known as the Data Withholding Attack. If η is unconstrained, a major mining pool X controlling a substantial portion of the network hash rate (e.g., 45%) can intentionally withhold the full data of the $n-th$ block while broadcasting only the compact block. This forces the remaining 55% of honest but uninformed miners to broadcast False declarations out of necessity. Under a setting where η = 4, the honest majority's effective hash rate is severely attenuated from 55% to a mere 13.75% (55/η). Consequently, the malicious pool X's effective dominance artificially surges to approximately 76.3%, granting the withholding pool an unintended 51% monopoly over the $n+k$ block space to capture monopolistic rewards.

Second, a hyper-punitive η inadvertently threatens systemic resilience during genuine network anomalies. In rare instances where the $n-th$ full block is legitimately corrupted or completely isolated due to global telecommunication failures, the honest majority must intentionally and correctly construct a False declaration chain to preserve the ledger's continuity. If η is tuned too high, the honest nodes attempting to recover the network would be subjected to prohibitive computational overhead. A moderately bound η minimizes the operational and financial burden on honest nodes during such necessary network recovery periods, preventing prolonged degradation of network liveness.

To satisfy both boundary conditions, η must be bound such that the honest fraction (0.55), even when forced into a temporary False status by a withholding pool or an infrastructure failure, retains an effective dominance greater than 50% to maintain network safety. Assuming a maximum plausible single-pool hash concentration of $H_{max} = 0.45$ (45%):

$$\frac{\frac{1 - H_{max}}{\eta}}{\frac{1 - H_{max}}{\eta} + H_{max}} > 0.5 \Longrightarrow \eta < \frac{1 - H_{max}}{H_{max}} \approx 1.22$$

Therefore, the theoretical upper bound is established at η < 1.22. To maximize the deterrent impact on strategic defectors while mitigating the operational burden during authentic network disruptions and preventing whale manipulation, $\eta = 1.2$ stands as the optimal equilibrium parameter for the protocol.

Crucially, the enforcement of $\eta = 1.2$ introduces an inherent self-stabilizing property during active chain splits (forks). Once a structural fork is initiated due to a transient informational asymmetry regarding the $n-th$ block—manifesting as a competitive race between a True declaration branch and a False declaration branch—both factions are driven by an existential mandate to maximize their cumulative work gravity as rapidly as possible. Under this intense competitive pressure, rational miners on both sides are mathematically disincentivized from broadcasting any subsequent unwarranted False declarations for future blocks $n+i$ $(i>0)$. Devised tactics of deliberate misreporting in a subchain are economically disincentivized, as doing so would inject a difficulty multiplier onto their own path, significantly suppressing their branch extension velocity. Consequently, even within a hostile fork race, the declaration behavior for all subsequent blocks inherently and immediately converges back to truth telling. To choose defection under a fork condition lowers block survivability, rendering the strategy economically non-viable for rational miners.

4. SECURITY ANALYSIS

This section evaluates the security resilience and engineering efficiency of the proposed protocol. The analysis is based on the core principle: "Allow false negatives (misunderstandings) but architecturally block false positives (fraud)."

4.1. Fork Convergence and Resolution Analysis

The proposed protocol achieves a superior level of network stability compared to Bitcoin by decoupling the logical consensus process from physical data synchronization. This section demonstrates how our system resolves forks faster, cleaner, and with significantly less computational overhead than the traditional Bitcoin model.

4.1.1. Fork Convergence Equivalence and Structural Efficiency

The proposed mechanism does not fundamentally complicate fork dynamics compared to Bitcoin. Rather than redefining the convergence rule itself, the protocol primarily regulates the initial condition under which a fork emerges. The core convergence principle remains unchanged (See 3.4.2).

We distinguish two types of fork events in the proposed protocol:

- A-type fork: a fork induced by Compact Block propagation delay.
- B-type fork: a fork induced by inconsistency in the $n_k_declaration$ field.

We show that, in both cases, the post-divergence convergence process reduces to the same stochastic model as in Nakamoto consensus.

A. A-Type Fork (Compact Block Propagation Delay)

An A-type fork arises solely from bounded network latency in the dissemination of Compact Blocks. In this case, two competing Compact Blocks at the same height may be received by different subsets of miners, resulting in a temporary divergence.

However, this scenario is structurally identical to the classical fork condition in Bitcoin. The only difference lies in the payload format (Compact Block rather than Full Block), not in the consensus logic governing branch selection.

Once the divergence occurs, miners extend the branch they first observed, and subsequent convergence is determined exclusively by relative hash power under the longest-chain rule. No additional protocol-level state or validation rule influences branch selection in this case.

Therefore, the post-divergence evolution of an A-type fork reduces exactly to the stochastic longest-chain competition of Nakamoto consensus. Under the standard assumptions of honest majority and bounded network delay, the probabilistic convergence guarantees are unchanged.

In short, A-type forks are behaviorally equivalent to Bitcoin forks and require no additional security analysis beyond that already established for Nakamoto consensus.

B. B-Type Fork ($n_k_declaration$ Inconsistency)

To rigorously evaluate the security and fork dynamics of the proposed protocol, we formalize the global state space at the execution boundary of the $(n+k)-th$ block evaluation. The deterministic convergence of the protocol relies on restricting the theoretical state space into a minimal set of practically viable regimes via rational miner assumptions.

We model the protocol environment and miner behaviors using three state variables:

- $V \in \{0,1\}$: The objective validity of the mined block B_n.

• $A \in \{0, 1, 2\}$: The network-wide information availability state of the corresponding Full Block FB_n at the evaluation point, where 0 indicates no nodes have verified the data, 1 represents partial dissemination, and 2 denotes 100% propagation to honest nodes.

• $D \in \{0, 1, 2\}$: The aggregate declaration status $n_k_declaration$ on the active tips, where 0 represents unanimous 'False', 1 represents unanimous 'True', and 2 represents a split opinion (the occurrence of a B-type fork).

While the combinations yield 18 theoretical states $2 \times 3 \times 3$, the interaction between objective protocol rules and rational incentive constraints filters out non-viable regimes. This exhaustive state space reduction is systematically enumerated and evaluated in Table 4.

Table 4. Systematic State Space Reduction via Rational Strategy and Physical Constraints

Cases	V	A	D	Viable?	Regime / Classification	Elimination / Optimization Logic
1	0	0	0	Yes	Regime I: Invalid Inception	Unique viable state for V=0; symmetric η-penalty race.
2–3	0	0	1, 2	No	Eliminate (Incentive)	Rational miners cannot declare True for unverified data.
4–9	0	1, 2	0, 1, 2	No	Eliminate (Physical)	Invalid blocks are blocked by validation layer; A ≥ 1 is impossible.
10–12	1	0	0, 1, 2	No	Eliminate (Physical)	Creator miner always holds data; absolute A=0 is impossible for V=1.
13	1	1	0	No	Eliminate (Incentive)	Dominated strategy; miners with data will not choose False.
14	1	1	1	No	Eliminate (Incentive)	Dominated strategy; uninformed miners will not risk True.
15	1	1	2	Yes	Regime III: Asymmetric Fork	B-Type Fork Zone; Data Withholding or Net-Latency.
16	1	2	0	No	Eliminate (Incentive)	Rational nodes with 100% data will never declare False.
17	1	2	1	Yes	Regime II: Canonical Chain	Standard Nakamoto consensus; uncontested honest progress.
18	1	2	2	Yes	Regime II: Strategic Attack	Malicious false declaration boundary.

By establishing the operational and game-theoretic filters detailed above, we condense the comprehensive matrix into three viable consensus regimes:

1. Invalid Inception Zone ($\boldsymbol{V = 0, A = 0, D = 0}$):

Invalid blocks are immediately purged by honest verification layers ($A = 0$). All rational miners unanimously declare $n_k_declaration = False (D = 0)$ to avoid block orphaning, reducing the system back to a symmetric Nakamoto PoW race with uniform penalty distribution.

2. Uncontested Canonical Convergence ($\boldsymbol{V = 1, A = 2, D \in \{1, 2\}}$):

When the data is fully available ($A = 2$), the system naturally maintains an honest consensus ($D = 1$). If an adversary strategically misreports ($D = 2$), it invokes the economic isolation regimes analyzed in Theorem 2 and Theorem 3.

3. Asymmetric Information Forking ($V = 1, A = 1, D = 2$):

In cases of partial data availability ($A = 1$)—caused either by malicious data withholding (analyzed in Theorem4) or physical network propagation latency—a B-type fork is mathematically inevitable ($D = 2$). Rational miners with the data will select $n_k_declaration = True$ to capture rewards, while uninformed rational miners must select $n_k_declaration = False$ to avoid structural invalidation risks. Uncontested states ($D = 0$ or $D = 1$) under $A = 1$ are strictly dominated strategies and are thus economically eliminated.

A B-type fork arises when competing blocks at height $n + k$ contain inconsistent $n_k_declaration$s ($D = 2$) under the protocol's deterministic validation rule. Such inconsistency creates a divergence at the declaration boundary.

This work does not alter Bitcoin's longest chain based probabilistic convergence algorithm. Instead, it introduces structured control over the moment of fork initiation through the timing and verification constraints associated with the $n_k_declaration$ field.

While the $n_k_declaration$ rule serves as a validity filter at the fork boundary, it leaves the longest-chain extension rule used by honest nodes unchanged.

After the initial divergence point, the system reduces to the same stochastic convergence model as Nakamoto consensus.

In particular, the commonly stated intuition that "a length difference of one block leads to immediate convergence" remains valid only under the following assumptions:

- The longer chain is valid under the protocol rules.
- All Full Blocks declared as "True" are made available and verified.
- An honest majority (i.e., >50% of total hash power) ensures positive drift toward the honest chain.

These assumptions are identical to those required by Bitcoin's security model. Therefore, under the same honest majority and bounded network delay model as assumed in Bitcoin, the proposed protocol preserves the probabilistic fork convergence properties of Bitcoin, while modifying only the control logic governing the initial fork event. This relationship is formally established in the following theorem.

Theorem 1 (Asymptotic Fork Resolution via Reduction to the Classical Longest-Chain Problem)

Consider a blockchain protocol in which:

- Nodes that have received block B_n set $n_k_declaration = true$ (Group A),
- Nodes that have not received block B_n set $n_k_declaration = false$ (Group B).

Assume that no fork exists up to height $n + k - 1$, and that a fork arises at height $n + k$. Assume also that, for every block declared True, the corresponding Full Block data is eventually reachable by every honest node required to verify it (i.e., there is no permanent data withholding). The timing of this reachability, for the block whose declaration triggers the B-type fork, is refined by the following two assumptions.

Assumption 1. The data-availability asymmetry that gives rise to a B-type fork persists at the evaluation height $n + k$ but is fully resolved, for every honest node required to verify it, by the production of the block at height $n + k + 1$.

Assumption 2. No subsequent Full Block propagation delay shall induce a separate declaration inconsistency—thereby creating a nested or additional B-type fork—before the initial B-type fork has been resolved.

Let a_t and b_t denote the respective chain lengths of Group A and Group B extended beyond block $n + k - 1$. Define $a_z = min(a_t, b_t)$ and $m = |a_t - b_t|$.

Then, although the initial fork configuration differs from that of Bitcoin due to partial data availability in the interval $[n, n + k - 1]$, under the standard assumptions of the classical longest-chain analysis (honest majority and bounded propagation delay), the subsequent fork convergence process is resolved by the longest-chain rule as in Bitcoin, converging asymptotically to a single canonical chain once the system enters the rule-determined convergence regime. The proof establishes this convergence by reducing the B-type fork to the classical longest-chain convergence problem; it does not re-prove the latter.

Proof.

1. Mutually Exclusive Case Exhaustion (Threshold Convergence Regime)

At height $n + k$, the fork initiates under partial data availability restrictions. Unlike Bitcoin, where the state up to height $n + k - 1$ is globally consistent, this protocol allows asymmetric data availability regarding blocks $B_n, \ldots, B_{n+k-1}$, though

compact blocks are successfully propagated. This asymmetry determines the initial configuration through three mutually exclusive cases:

By Assumption 2, no subsequent Full Block propagation delay induces a separate declaration inconsistency before the initial B-type fork is resolved. Hence the cases below are mutually exclusive and exhaustive, and the fork can be represented by the two-group state variables a_t, b_t.

Case A (Group A mines first): Group A extends its chain beyond $n + k - 1$ ($a_t = 1, b_t = 0$).

- **Case A-1 (Delayed Balance):** If Group B successfully mines a competing block at height $n + k$ before Group A can discover the next block($n + k + 1$), geometric equality is restored: $(a_t, b_t) = (1,1)$, entering a balanced competitive state ($m = 0, a_z = 1$).
- **Case A-2 (Dominant Convergence via True Chain):** If Group A mines the subsequent block before Group B responds, the length difference widens ($m > 1, a_z = 0$), immediately routing the system into the Dominant Convergence Regime (Det 2).

Case B (Group B mines first): Group B extends its chain first ($a_t = 0, b_t = 1$), creating a temporary imbalance.

- **Case B-1 (Dominant Convergence via False Chain):** If Group B mines again before Group A can discover a block at height $n + k$, the difference satisfies $m > 1$ with $a_z = 0$, transitioning directly into the Dominant Convergence Regime (Det 2).
- **Case B-2 (Restored Balance):** Otherwise, Group A successfully discovers a block at height $n + k$, restoring symmetry: $(a_t, b_t) = (1,1)$, which reduces the outcome to the balanced state ($m = 0, a_z = 1$).

Case C (Long-Range Forks): In the case of prolonged network partition where a fork initializes with an immediate block differential satisfying $m > 1$ and $a_z \geq 0$, the structural variance of data availability is overridden by cumulative proof-of-work, bypassing the balanced phase to enter the Dominant Convergence Regime (Det 2).

2. Formulating the Rule-Determined Convergence Regimes

The term "deterministic" in the labels Det 1 and Det 2 refers only to the deterministic application of the selection rule, not to a deterministic outcome; the convergence outcome remains probabilistic under the longest-chain rule.

Once the system exits the threshold regime, its convergence behavior is governed by two distinct rule-determined regimes based on the state variables m and a_z:

Frictional Convergence Regime (Det 1: $\boldsymbol{m = 1, a_z > 0}$): A single-block lead exists. While minor propagation delays (friction) occur during reorgs, convergence is guaranteed by the longest-chain rule as a_z scales.

Dominant Convergence Regime (Det 2: $\boldsymbol{m > 1}$): A lead of two or more blocks completely neutralizes any past data withholding noise, forcefully compelling a single canonical chain resolution.

3. State Space Reduction and Stochastic Transition

The sole non-deterministic outcome from the threshold regime is the balanced state ($m = 0, a_z = 1$). Under the condition where $a_z > 0$, the evaluation stage dictated by Section 3.4.2.A (Threshold Convergence where $m = 1$ and $a_z = 0$) has already been bypassed. Consequently, any subsequent differential in chain length triggers a transition into the rule-determined regime, allowing the mechanics of this process to be formalized as follows.

Assumption 2 also justifies reducing the balanced state to the one-dimensional Markov chain below; without it, nested B-type forks would introduce additional declaration groups and invalidate this reduction.

To model its evolution, we define the stochastic state variable:

$$\Delta_t := a_t - b_t$$

At the balanced configuration, the initial boundary condition is established as $\Delta_0 := 0$. Each subsequent block discovery updates the system state via a one-dimensional Markov chain:

$$\Delta_{t+1} = \begin{cases} \Delta_t + 1 & \text{with probability } \alpha, \\ \Delta_t - 1 & \text{with probability } 1 - \alpha, \end{cases}$$

where α represents the mining power ratio of Group A. The stopping time for this process is defined as:

$$\tau = inf\{t : |\Delta_t| = 1\}$$

Since $\Delta_0 = 0$ and each step is ± 1, we have $\tau = 1$ almost surely. The stopping time τ therefore marks the first moment at which one group leads by a single block. It is a formal stopping boundary of the Markov chain, not an absorbing state of the underlying chain.

The relevant event for the protocol is not the sign of the drift but the occurrence of $|\Delta_t| = 1$. At each such occurrence, the fork selection rule becomes determinate: the competing branches are distinguishable, and a unique chain can be selected according to the rules of Section 3.4.2. This is the sense in which the balanced state is exited.

The behavior of Δ_t under this Markov chain is as follows. If $\alpha > 1/2$, each step is more likely to be +1 than −1; if $\alpha < 1/2$, each step is more likely to be −1 than +1; and if $\alpha = 1/2$, the two directions are equiprobable. The sign of the drift therefore determines only which group prevails when a divergence occurs, not whether $|\Delta_t| = 1$ occurs or whether unique chain selection is possible at such events. At every occurrence of $|\Delta_t| = 1$, the fork selection rule remains determinate and a unique chain can be selected. Theorem 1 requires only the occurrence of $|\Delta_t| = 1$, at which point the fork selection rule becomes determinate and a unique canonical chain can be selected.

The parameter α in the reduced Markov chain denotes the instantaneous hash-power share of Group A at the moment of divergence, not a time-invariant quantity. Once $|\Delta_t| = 1$ occurs and a candidate chain becomes identifiable under the selection rule, honest hash power reallocates toward that chain. Convergence is therefore driven not by the sign of the drift but by this self-reinforcing chain-selection feedback. The Markov chain models only the short pre-selection window; the subsequent dynamics reduce to the classical longest-chain process. This mirrors the standard treatment of Bitcoin, where αα is likewise held fixed only for the duration of the fork race.

Upon reaching the stopping boundaries at ± 1, the system exits the balanced state and transitions into the Frictional Convergence Regime (Det 1), where the fork selection rule is determinate. Det 1 and Det 2 are not absorbing states; they are the regimes in which a unique canonical chain can be selected by the longest-chain rule. The term "convergence" in the labels Det 1 and Det 2 refers to this determinacy of selection, not to an irreversible absorption.

Note. As formalised in the operational rules of Section 3.5, any rational mining faction attempting to propagate a fork is incentivized to preserve their branch extension velocity. Consequently, following the initial defection, they are economically incentivized to immediately revert to "True" declarations for all subsequent valid blocks $n + i (i > 0)$ to avoid the difficulty multiplier. Because rational actors universally choose to bypass this penalty zone to maximize their block survivability, their effective hash power fraction effectively returns to the original constant $1 - \alpha$ rather than being throttled to $(1 - \alpha)/\eta$. Therefore, the state transition probabilities within this Markov chain remain fundamentally governed by the static hash power distribution, rendering the explicit dynamic integration of η within the state space reduction unnecessary.

4. Asymptotic Convergence via Dynamic Validation Relaxation

Whether by immediate routing through structural dominance (Cases A-2, B-1, and C) or through the stochastic stopping boundary of the reduced Markov chain ($m = 0 \rightarrow$ Det 1), every trajectory reaches a state in which the fork selection rule is determinate.

Within this regime, once a chain obtains a length advantage ($m \geq 1$), the validation rules are dynamically enforced based on block declarations: while blocks declared as "True" strictly undergo complete data verification, blocks declared as "False" are universally accepted without requiring full data. By Assumption 1, the Full Block data FB_n required to verify the True

declaration of the initial B-type fork is available to every honest node required to verify it by the production of block $n + k + 1$. Together with Assumption 2, which ensures that no additional True declaration from a nested B-type fork must be verified, this makes the verification-based transition to the winning True chain feasible. This operational mechanism completely nullifies the impact of earlier data availability asymmetries over the interval $[n, n + k - 1]$, ensuring that internal differences in data withholding no longer influence the stochastic evolution of the fork. Consequently, the long-term fork resolution mechanics reduce to a competition of cumulative proof-of-work weight, guaranteeing that the network asymptotically converges to a single canonical chain via the longest-chain rule. □

This completes the reduction. The balanced state is exited in finite time almost surely, and the subsequent evolution from Det 1 and Det 2 is exactly the classical longest-chain convergence problem, which the classical analysis resolves under its standard assumptions (honest majority and bounded propagation delay). Theorem 1 invokes that result rather than re-proving it; which chain prevails is a separate security question addressed in Sections 3.5 and 4.2. A transient violation of Assumption 1 does not induce systemic deadlock or collapse: any temporary data asymmetry merely extends the fork depth until the missing Full Block arrives, after which the classical longest-chain dynamics resume; in the extreme case where the data never arrives, the True branch fails validation and honest nodes proceed along the False branch, temporarily stalling finality while preserving liveness.

Remark 1.1 (Scope and Status of the Following Relaxation)

The proof of Theorem 1 above proceeds under Assumption 1, which requires that the data-availability asymmetry giving rise to a B-type fork be fully resolved by height $n + k + 1$. The preceding paragraph examined the complementary case in which this condition fails—either because the asymmetry persists beyond the k −window or because the Full Block never arrives—and argued that such a failure does not induce systemic deadlock or collapse. The present remark and the proposition that follows address the intermediate regime that lies between these two extremes: the asymmetry persists beyond $n + k + 1$ but is resolved at some height $n + k + j$ with $1 < j < k$.

This supplement does not alter Theorem 1 as stated. Theorem 1 remains valid under Assumption 1, and its proof is unchanged. The purpose of Proposition 1 is to establish a robustness extension of Theorem 1: the same convergence conclusion holds under a strictly weaker condition, provided that no nested B-type fork intervenes during the extended asymmetric interval. Accordingly, the conditions below are labeled Relaxed Conditions rather than Assumptions, and the result is stated as a Proposition rather than a Lemma, so that it is not misread as a premise on which Theorem 1 depends.

Relaxed Condition R1 (Bounded Data-Availability Asymmetry). The data-availability asymmetry that gives rise to a B-type fork persists at the evaluation height $n + k$ but is fully resolved, for every honest node required to verify it, by the production of the block at height $n + k + j$, where $1 < j < k$.

Relaxed Condition R2 (No Nested B-Type Fork Before Resolution). No subsequent Full Block propagation delay shall induce a separate declaration inconsistency—thereby creating a nested or additional B-type fork—during the interval $[n + k, n + k + j]$, i.e., before the initial B-type fork has been resolved.

Definition (Bitcoin's j −Block Complete-Partition Model). Bitcoin's j −block complete-partition model is a reference model of Nakamoto consensus in which the Bitcoin network undergoes a temporary physical partition. Let h_0 denote the height of the common ancestor at which the two partitions diverge. The network is divided into two disjoint physical partitions at height h_0, and no block or message crosses the partition boundary while the partition persists. Mining proceeds independently within each partition according to the hash power available in that partition, under Bitcoin's difficulty-adjustment rule; j is assumed to be small enough that no difficulty retarget occurs during the partition interval, so the difficulty is constant over that interval. On a global timeline, the partition is assumed to persist until the moment at which one partition first reaches height $h_0 + j$, i.e., first produces a block at height $h_0 + j$; at that moment the physical partition is lifted and communication between the two partitions is restored. The other partition may be at height $h_0 + j - 1$ or lower at that moment. Upon restoration, the two partitions exchange their chains, and the canonical chain is selected by Bitcoin's longest-chain rule, i.e., the chain with the greatest cumulative proof-of-work prevails; under constant difficulty, this coincides with the greater chain length. The shorter chain is discarded and its blocks are orphaned. If both partitions produce a block at height $h_0 + j$ simultaneously on the global timeline, the tie is resolved by Bitcoin's tie-breaking rule, i.e., the block received first is preferred. The convergence time of this model is the time required, from the moment the partition is lifted,

for all honest nodes in both partitions to adopt a single canonical chain, including the propagation and validation of the winning chain's missing blocks under Bitcoin's standard network assumptions.

Proposition 1 (Robustness of Theorem 1 under Relaxed Conditions R1 and R2)

Assume Relaxed Conditions R1 and R2. Consider a B-type fork initiated at height $n+k$. On the interval $[n+k, n+k+j]$, the process is an asymmetric longest-chain process with an absorbing boundary for the False branch and a verification-gated boundary for the True branch. At $n+k+j$, the data-availability asymmetry vanishes, and the merge-state space of the process is contained in that of Bitcoin's j −block complete-partition model. The proposition asserts a reduction at the level of merge states; where reduction means the inclusion relation established below; it does not assert a comparison of convergence times, nor equality of the probability distribution over which branch prevails.

Proof.

At $a_z = 0$, the initial B-type fork can be modeled as a virtual physical partition: the group that has received the Full Block (Group A) and the group that has not (Group B) do not accept each other's blocks, so mining competition at equal height is structurally identical to Bitcoin mining under a physical partition. Once $max(m-1, a_z) > 0$, the two groups observe asymmetric boundaries. From Group B's perspective, the network appears fully connected, because a False branch that leads by at least one block is accepted network-wide without Full Block data. From Group A's perspective, the network still appears partitioned according to Full Block dissemination, and the True branch is extended only by the data-holding set D_t. As a_z increases, D_t expands monotonically and eventually covers the whole network. Therefore, the interval $[n+k, n+k+j]$ is an asymmetric longest-chain process, and its merge states reduce to those of Bitcoin's j −block complete-partition model.

Reduction to Bitcoin's j −block complete-partition model. Let $L_T(h)$ and $L_F(h)$ denote the lengths of the True and False chains at absolute height h, so that $L_T(n+k) = L_F(n+k) = n+k$. Suppose the process reaches height $n+k+j$ without prior resolution. If the False branch had ever led by at least one block before $n+k+j$, Section 3.4.2.B would have resolved the fork at that moment, so the process could not have reached $n+k+j$.

Hence on $[n+k,\ n+k+j)$ the True branch is never behind:

$$L_T(h) \ \geq \ L_F(h) \ for \ all \ h \in [n+k,\ n+k+j).$$

At height $n+k+j$, R1 restores full data availability, so the merge state is determined solely by the length difference

$$m' = L_T(n+k+j) - L_F(n+k+j)$$

and, when $m' = 0$, by which branch produced the block at $n+k+j$ first. The inequality gives $L_T(n+k+j-1) \geq L_F(n+k+j-1)$; once the block at $n+k+j$ is produced, the length difference satisfies $m' \geq -1$. Three cases exhaust all possibilities:

1. False produces the $n+k+j-th$ block first. By the same inequality, this is possible only if the chains were tied immediately before that block. False then leads by one, and by Section 3.4.2.B the network converges to the False chain. This is exactly Bitcoin's j −block complete-partition merge state with a one-block False lead.

2. True produces the $n+k+j-th$ block first. Then the True branch leads by $m' \geq 1$ at the merge point (if it already led by $m' \geq 1$ before this block, the block at $n+k+j$ is already a True block, and the lead is at least m'). This is Bitcoin's merge state with a True lead of $m' \geq 1$.

3. Both branches produce the $n+k+j-th$ block simultaneously. Then $m' = 0$ at the merge point, which is Bitcoin's merge state with equal chain lengths, resolved by the same tie-breaking rule.

Thus every possible state at $n+k+j$ maps to a merge state of Bitcoin's j −block complete-partition model. The only Bitcoin merge state not realized is a False lead of two or more blocks ($m' \leq -2$); the bound $m' \geq -1$ established above excludes it.

What this establishes is an inclusion relation at the level of merge states:

$$MergeStates_{\Pi}(n+k+j) \subseteq MergeStatesBitcoin_{j-partition}.$$

That is, the proof shows that the merge states at $n+k+j$ form a subset of the merge states of Bitcoin's j −block complete-partition model. This subset relation suffices for the reduction. The convergence theorem for Bitcoin's j −block complete-

partition model is a universal statement: from any merge state, the longest-chain rule yields convergence. If every merge state of the proposed protocol is contained in that set, the same theorem transfers directly to the proposed protocol. A superset relation would not suffice, since it would admit states outside the Bitcoin model for which the theorem has not been established. The subset relation is therefore the minimal and correct direction required for the reduction.

Two conditions must hold for the reduction to be complete: (a) the inclusion relation between merge states, and (b) identity of the post-merge dynamics. Condition (a) has been established above. Condition (b) follows because R1 restores data availability at $n + k + j$ and R2 excludes nested B-type forks, so that the residual process is the symmetric Nakamoto longest-chain process and the post-merge transition rule is identical to that of Bitcoin.

With the merge state thus reduced, at $n + k + j$ R2 ensures that no nested B-type fork has intervened; the asymmetric boundaries therefore disappear, and absent further B-type forks the residual process is exactly the symmetric Nakamoto longest-chain process. Accordingly, on the interval $[n + k, n + k + j]$, the protocol reduces, at the level of merge states, to Bitcoin's $j -$ block complete-partition model, and it introduces no convergence mechanism beyond Bitcoin's. The proposition asserts a reduction of the merge-state space and does not assert a comparison of convergence times. This proves the proposition. □

Under the original Assumption 1, the asymmetric interval collapses to a single height $n + k + 1$, and Proposition 1 reduces to the case $j = 1$. Theorem 1 is therefore the boundary case of Proposition 1, and the two results are consistent.

Remark 1.2 (Relationship to Bitcoin's Operating Assumptions).

The assumptions underlying Theorem 1 are not specific to the proposed protocol. Rather, they are the pipeline analogue of the operating conditions under which Nakamoto consensus itself is proven secure. Bitcoin's convergence guarantees — common prefix, chain quality, and chain growth — hold only under the assumption that the block propagation delay δ is small relative to the mining interval λ^{-1}. When δ becomes comparable to or exceeds λ^{-1}, fork depth increases and the classical security proofs no longer apply. Assumption 1 generalizes this delay bound from a single block interval to the $k -$block declaration window: the condition $T(B_n) \leq (k - i) \cdot \lambda^{-1}$ plays the role that $\delta \ll \lambda^{-1}$ plays in Bitcoin. Assumption 2 generalizes the shallow-fork consequence of that bound to the declaration dimension: it requires that nested B-type forks be negligible, just as Bitcoin requires that nested forks be negligible with high probability. In this sense, Theorem 1 does not introduce a qualitatively stronger security requirement than Nakamoto consensus; it makes these requirements explicit and adapts them to the pipelined structure. As with Bitcoin, Assumptions 1 and 2 are calibrated as parameter-dependent probabilistic guarantees (Section 5.2.2).

4.1.2. Structural Advantages in Fork Resolution

A. Agility of Convergence: Compact Block Driven Resolution

In Bitcoin, the convergence of a fork is strictly bound to the propagation speed of Full Blocks, which often exceed several megabytes. This makes fork resolution sensitive to full-block propagation latency. In contrast, our protocol utilizes Compact Blocks consisting only of headers and compressed transaction IDs as the primary vehicle for fork resolution.

Comparison: While a several MB Bitcoin block may experience non-negligible propagation delay across a globally distributed network, our KB sized Compact Blocks propagate significantly faster due to their reduced size.

Result: Fork resolution is effectively determined by the propagation of Compact Blocks carrying valid PoW headers. Given their significantly smaller size, the network identifies the "winning chain" within 1-2 blocks (PoW statistical average) almost without friction. This ensures that the fork state is ephemeral, whereas forks in Bitcoin are prolonged by propagation delays of full blocks.

B. Network Hygiene: PoW Gated Full Block Propagation

Bitcoin's forks are "polluted" because conflicting Full Blocks from all competing chains are broadcast simultaneously, wasting global bandwidth on data that will eventually be discarded (orphan blocks). Our protocol implements the Strategic Propagation Delay (Section 3.2.4), which mandates that an $n\text{-}th$ Full Block is only broadcast after its $(n + i)\text{-}th$ Compact Block confirmation (typically $i = 2$).

Comparison: Bitcoin nodes may temporarily store competing branches until convergence. In our system, fork competition is primarily resolved at the Compact Block layer. In most cases, the dominant chain is statistically established before Full Block propagation.

Result: Only the Full Block belonging to the winning chain ever saturates the network. This "winner-takes-all" propagation model significantly reduces bandwidth overhead associated with losing branches. Consequently, this design reduces unnecessary full-block propagation in transient fork scenarios, at the cost of introducing a bounded confirmation delay.

C. Computational Simplicity: On/Off Atomic Switching via Pre-verified Data Sharing

Suppose a fork occurs at the $(n+k)-th$ block, splitting the network into Group A and Group B, and is subsequently resolved by converging to Group A at the $(n+k+i)-th$ block $(0<i<k)$. Under this scenario, the fork resolution represents the optimal synergy between the proposed protocol's 'Delayed Finality' and 'Data Availability.' The following analysis evaluates the efficiency of this process in comparison to the legacy Bitcoin model.

Structural Resilience through Atomic Chain Reorganization ($O(1)$ Switching)

Bitcoin: When a winning chain is determined at height $(n+k+i)$, the losing group (Group B) faces critical Physical Bottlenecks. Group B must download every Full Block from the fork point $(n+k)$ to the current tip $(n+k+i)$ from Group A. In a large-block environment, this bandwidth consumption significantly delays recovery. Post-download, nodes must sequentially re-verify every transaction within that interval to update the local UTXO Set.

Proposed Protocol: The proposed scheme simplifies the recovery process through Latency Hiding. In the proposed scheme, if a fork occurs from the $(n+k)-th$ block, Group A and Group B will also have entirely different blocks from $n+k$ to $n+k+i$, similar to Bitcoin. However, the Compact Blocks from n to $n+k-1$ have already been shared. During the chain reorganization, Group B nodes only need to download the Compact Blocks from $n+k$ to $n+k+i$ of Group A. Provided that the condition is guaranteed where Full Blocks from the $n-th$ to the $(n+i)-th$ are already locally available via background propagation (See Section 5.2.2), nodes can instantly shift the chain tip simply by identifying and linking these blocks. Consequently, the Reorg is not a physical data migration but a logical switching operation(On/Off Switching) that activates the 'status' of data already residing in the local buffer.

Convergence to Physical Baseline under Macro-scale Disruptions

The observation that "long-term network partitioning causes the system to behave similarly to Bitcoin" describes a crucial 'Safety Boundary' within the same scenario.

Boundary Condition ($i \geq k$ or Propagation Failure): If the fork depth exceeds the $k-$window or if background propagation is impaired by network failure—preventing Latency Hiding from functioning—nodes cannot procure Full Block data asynchronously.

Convergence to Baseline: In such cases, the proposed protocol must also download winning chain data in bulk from scratch. This is a Physical Baseline, an inherent consequence for all PoW algorithms governed by the CAP Theorem.

Result: The fundamental advantage lies in maximizing resilience against 'routine and stochastic forks' to enhance network liveness, rather than attempting to transcend physical laws.

4.1.3. Effect of i-Confirmation Gated Propagation on Fork Convergence

A. Independence of Gated Propagation from Direct Fork Resolution

The i-confirmation gated propagation mechanism (A node delays propagation until confirmation $depth \geq i$.) does not directly influence fork convergence in the proposed consensus protocol. The primary reason is that fork resolution is determined by the ordering of Compact Blocks rather than by the propagation timing of the corresponding Full Blocks.

First, fork convergence is governed by the most recently mined Compact Blocks that define the canonical chain ordering.

Since the Compact Blocks are propagated independently of the associated Full Block data, the immediate convergence behavior of forks is largely unaffected by whether the corresponding Full Block has already propagated through the network.

Second, the final validation decision affecting consensus is determined by the Full Block that was mined k blocks earlier.

Because this validation event concerns blocks that are already sufficiently deep in the chain, it is generally independent of the propagation timing of the most recently mined Full Block.

Nevertheless, the i-confirmation gated propagation mechanism may still exert an indirect influence on the system. By design, it delays the network-wide propagation of newly mined Full Blocks by approximately i block intervals. Although this delay

does not affect the immediate fork-resolution process, the propagated Full Block will eventually participate in the validation decision once an additional k blocks have been mined. In that sense, the delayed propagation may indirectly influence the later validation phase of the protocol.

Therefore, the i-confirmation gated propagation mechanism should be interpreted primarily as a network-level stabilization mechanism rather than as a direct component of the fork-convergence rule.

B. Interaction Between i-Confirmation Gated Propagation and Universal Propagation

The effectiveness of the proposed i-confirmation gated propagation mechanism becomes more evident when it operates together with the Universal Propagation principle described in Section 3.2.2. Consider the following scenario involving a temporary network disruption.

Assume that when the $n + k - th$ block is mined, the Full Block F_n has propagated to only about 55% of the nodes due to a severe network disturbance. Let this subset of nodes be denoted as group A, while the remaining nodes that have not yet received F_n form group B.

Because nodes in group A have already received the Full Block F_n, they will mine blocks with $n_k_declaration = true$.
In contrast, nodes in group B will mine blocks with $n_k_declaration = false$. As a result, the network temporarily splits into two competing branches.

During this period, the i-confirmation gated propagation mechanism prevents newly mined Full Blocks from being immediately disseminated across the network. Consequently, neither group can quickly propagate their Full Blocks to the other partition, and the fork evolution is primarily determined by the sequence of Compact Blocks. Now consider the moment when the $n + k + z - th$ block $(z \geq 1)$ is mined.

If group A mines this block first, the Universal Propagation property implies that there is a significant probability that the Full Block F_n has already reached a substantial portion of nodes in group B during the intervening time. In this case, the effective dissemination window for the Full Block expands from the conservative $k - i$ interval assumed in the security analysis to approximately $k - i + z$ blocks. Once this extended window reaches$(z = i)$ or exceeds $k(z > i)$, nodes across the network are likely to converge toward the chain containing the valid declaration.

A similar situation may arise even if group B mines the $n + k + z - th$ block first. Because Universal Propagation continuously spreads previously mined blocks across the network, the Full Block F_n may already have reached many nodes in group B before the fork is resolved. Of course, in cases where the entire network converges toward the branch maintained by group B, the dissemination of F_n holds no significance for the consensus outcome.

Consequently, the $k - i$ dissemination window used in the security analysis can be interpreted as a conservative lower bound.

In practice, due to Universal Propagation and the delayed dissemination introduced by i-confirmation gating, the effective propagation window may extend to approximately $k - i + z$ blocks, where $z \geq i$. This observation suggests that the proposed mechanism may provide stronger practical robustness than what is implied by the conservative analytical bound (See Section 4.3.2), as the continued dissemination of previously mined blocks can dynamically extend the effective propagation window. As a result, the proposed protocol exhibits an adaptive propagation tolerance, where the minimum dissemination window required for security is bounded by $k - i$, while the effective propagation window may become larger in practice due to the continued dissemination of previously mined blocks.

4.2. Mitigation of Attack Vectors and Resilience Analysis

This section analyzes the system's resilience against malicious attempts to disrupt the network by exploiting the asynchronous nature of data propagation and declaration.

4.2.1. Resilience Against Selfish Mining: A Formal Analysis

By structurally decoupling PoW consensus competition from Full Block data propagation, the proposed protocol ensures that the profitability of selfish mining is not augmented relative to the traditional Nakamoto Consensus.

In classical Nakamoto style blockchains, an adversary can obtain a strategic advantage by withholding a newly mined Full Block. During the withholding period, honest miners continue to extend a stale tip, thereby wasting hash power and enabling the attacker to increase the probability of achieving a private lead. The effectiveness of such a strategy depends critically on propagation delay and the associated propagation advantage parameter (γ), which governs tie-breaking outcomes during fork races.

Let $\alpha \in (0,1)$ denote the adversary's relative hash power and $\gamma \in [0,1]$ the probability that honest miners adopt the adversary's block in a race condition. The expected revenue of selfish mining can be expressed [23] as a function

$$R_{SM}(\alpha,\gamma) = \frac{\alpha(1-\alpha)^2\big(4\alpha+\gamma(1-2\alpha)\big)-\alpha^3}{1-\alpha(1+(2-\alpha)\alpha)}$$

and selfish mining becomes profitable if

$$R_{SM}(\alpha,\gamma) > \alpha.$$

Thus, profitability is fully determined by the pair (α,γ).

A. Decoupling Consensus from Payload Propagation

In the proposed design, consensus competition is determined exclusively at the Compact Block layer. A Compact Block contains only minimal metadata (e.g., block headers and transaction input identifiers), and its propagation delay is therefore dominated by network latency rather than payload size. Consequently, the additional propagation asymmetry attributable to Full Block transmission is substantially reduced.

Importantly, the timing of Full Block delivery does not influence the establishment of PoW consensus priority. Broadcasting a Compact Block while withholding the corresponding Full Block does not confer a private lead, because honest nodes immediately commence mining upon receipt of the Compact Block. Moreover, if the Full Block is not delivered within the prescribed $k-$window, the block is deterministically declared invalid ("False") at height $n+k$, resulting in complete forfeiture of the associated block reward. Thus, Full Block withholding cannot increase the adversary's expected revenue.

Accordingly, the expected payoff of selfish mining under the proposed protocol remains governed by the classical parameters α and γ, without introducing any additional gain derived from payload-level manipulation. In other words, the expected payoff structure of selfish mining remains bounded by the same $\alpha-\gamma$ conditions established in Nakamoto Consensus.

Under the assumption that Compact Block propagation is not slower, nor more asymmetric, than Full Block propagation in the Bitcoin network (See Section 4.3.2), the proposed protocol does not reduce the threshold condition for profitable selfish mining.

B. Granular Analysis of Lead-Reduction Disclosure ($\Delta \rightarrow \Delta-1$)

We now analyze the canonical selfish mining maneuver at finer granularity.

In classical selfish mining, the attacker maintains a private lead

$$\Delta = L_{adv} - L_{honest} \geq 1.$$

When the honest chain mines a new block and reduces the lead by one (i.e., $\Delta \rightarrow \Delta-1$), the attacker strategically releases one previously withheld block to trigger a fork race. The success of this maneuver depends on γ, which determines the fraction of honest miners that follow the attacker's branch in the resulting tie.

Under the proposed protocol, the effect of this disclosure depends critically on whether the attacker's privately mined block carries an $n_k_declaration$ consistent with the honest majority.

Case 1: Declaration Alignment

If the attacker's block carries an $n_k_declaration$ consistent with the collective decision of the honest majority, disclosure induces a fork race analogous to the classical model. However, since consensus priority is determined at the Compact Block layer and Compact Block propagation is not assumed to be more asymmetric than Bitcoin's Full Block propagation,

$$\gamma_{new} \leq \gamma_{bitcoin}.$$

Thus, even in the canonical $\Delta-$reduction disclosure scenario, the attacker's strategic advantage is not increased relative to the classical setting.

Case 2: Declaration Misalignment

If the attacker's withheld block carries an $n_k_declaration$ inconsistent with the honest majority's validated state, then upon disclosure the block fails declaration validation and is rejected at the consensus boundary. Honest nodes refuse to extend that branch, and no effective tie-breaking race occurs.

For that disclosure event, the effective propagation advantage collapses:

$$\gamma_{new} \approx 0.$$

Hence, the canonical "lead reduction → strategic release" maneuver is either bounded by classical γ (alignment case) or neutralized at the validation layer (misalignment case).

Threshold Preservation

In both cases,

$$\gamma_{new} \leq \gamma_{Bitcoin}.$$

Let α^* satisfy

$$R_{SM}(\alpha^*, \gamma) = \alpha^*.$$

Since the propagation advantage parameter is not increased under the proposed protocol, the profitability threshold is weakly preserved:

$$\alpha^*_{new} \geq \alpha^*_{Bitcoin}.$$

Therefore, the protocol maintains, at minimum, the same level of resilience to selfish mining as Nakamoto Consensus, while eliminating any auxiliary advantage arising from Full Block withholding or payload-level propagation asymmetry.

4.2.2. Structural Disadvantage of False $n_k_declaration$ Attacks

We analyze the scenario in which an adversary deliberately falsifies the $n_k_declaration$ field. For concreteness, assume that the network is currently mining the block at height $n + k$. The adversary mines block with a declaration opposite to the honest majority's state by declaring "False" when the $n - th$ block has been widely propagated, or "True" when it has not. We assume the adversary does not declare "True" for a non-existent block. We compare this attack surface with classical Bitcoin.

Immediate Network Unification in Bitcoin: In Bitcoin, if an adversary mines the $n + k - th$ block first, the network can be unified under the adversary's chain simply by publishing that block. Since consensus is determined solely by longest-chain PoW priority, the first valid extension at height $n + k$ can immediately redirect honest mining power toward the attacker's branch. Thus, in Bitcoin, winning a single block race at height $n + k$ is sufficient to unify the network under the attacker's chain.

Structural Inability to Unify in the Proposed Protocol (Game-Theoretic and Mathematical Analysis of Consensus Stability): In the proposed protocol, this property no longer holds under "False" $n_k_declaration$. Even if the adversary mines the $n + k - th$ block first, the block carries a declaration inconsistent with the honest majority's validated state. Consequently, honest nodes reject the block at the declaration-validation layer and refuse to extend the attacker's branch. Therefore, mining a single block at height $n + k$ does not suffice to unify the network. This can be formalized as follows.

Theorem 2 (Economic Resilience to Malicious Declarations).

Assume a set of rational miners who possess the common knowledge that the $n - th$ block is valid and has already been propagated to over 90% of the nodes in the network ($p \geq 0.9$). Under this condition, any $n + k - th$ compact block carrying a malicious false declaration ($n_k_declaration = false$) will be ignored by these rational miners. Consequently, the honest chain remains the unique Nash Equilibrium, and the malicious branch will be orphaned with a probability of at least 0.81.

Proof.

Let B be the fixed block reward and m be the relative hashing power of an individual rational miner i. An attacker (minority $q \leq 0.1$) publishes a $n + k - th$ block with a malicious declaration ($n_k_declaration = false$), gaining a $z = 1$ block lead. Knowing that $p \geq 0.9$ of the network's hash power is already aligned with the valid $n - th$ block, miner i evaluates two strategies: Maintenance ($\boldsymbol{E}[\boldsymbol{G_T}]$) or Defection ($\boldsymbol{E}[\boldsymbol{G_F}]$).

1. Expected Utility of Maintenance ($\boldsymbol{E}[\boldsymbol{G_T}]$) :

Miner i recognizes that the honest majority, possessing $\geq 90\%$ of the total hash rate, will attempt to reorganize the chain. The probability $P_{success}$ of the honest majority mining two successive blocks ($n + k$ and $n + k + 1$) to reclaim the lead is:

$$P_{success} = p^2 \geq (0.9)^2 = 0.81$$

The expected gain for miner i on this chain is:

$$E[G_T] = m \cdot B \cdot P_{success} \geq 0.81 \cdot m \cdot B$$

2. Expected Utility of Defection ($\boldsymbol{E}[\boldsymbol{G_F}]$):

If miner i defects to the malicious branch, the reward is realized only if the 90% honest majority fails to reorganize. The probability of this failure is at most $(1 - P_{success})$.

Thus, the expected gain is:

$$E[G_F] = m \cdot B \cdot (1 - P_{success}) \leq (1 - 0.81) \cdot m \cdot B = 0.19 \cdot m \cdot B$$

3. Nash Equilibrium Condition:

Given the miners' knowledge of the 90% propagation, they compare the utilities:

$$E[G_T]/E[G_F] \geq 0.81 \cdot m \cdot B/0.19 \cdot m \cdot B \approx 4.26 > 1$$

Since the expected return for remaining on the true chain is significantly higher, and miners know their peers are also rational and informed, S_T becomes the only logical choice.

4. Convergence:

The shared knowledge of 90% propagation ensures that rational miners do not fear isolation on the true chain. Instead, they fear the 0.81probability of their block being orphaned if they join the false branch. Therefore, the false declaration is collectively ignored.

Note: As discussed in the proof of Theorem 1, the difficulty penalty η acts exclusively as an initial entry barrier at the fork initiation step. Since this proof assumes that the attacker has already established a $z = 1$ block lead, any subsequent block extensions within both branches must carry "True" declarations to avoid the penalty. Therefore, η does not trigger for future blocks, and its explicit integration is omitted in the utility analysis of Theorem 2. □

A critical question arises in the implementation of the aforementioned theorem: How can an individual node in a decentralized network attain high confidence that the $n - th$ block has indeed reached the 90% threshold? In a distributed system without a global observer, establishing such a consensus on propagation state is non-trivial. However, the proposed protocol provides a robust empirical mechanism for this verification: the analysis of the arrival interval between block signals.

To preserve the structural simplicity of the Proof-of-Work (PoW) mechanism, the protocol avoids hard-coded temporal constraints. Instead, it utilizes the arrival interval $\Delta T = T_{CB(n+k)} - T_{FB(n)}$, where $T_{CB(n+k)}$ is the arrival timestamp of the $n + k - th$ compact block and $T_{FB(n)}$ is the arrival timestamp of the $n - th$ full block, as a critical heuristic for individual nodes to estimate the network-wide propagation state. If ΔT significantly exceeds the known network diameter (the time required for a block to traverse the majority of the network), a miner attains high confidence that the $n - th$ block has achieved Common Knowledge among the 90% majority.

The proposed $n_k_declaration$ mechanism establishes a robust incentive structure that ensures systemic integrity through the principle of "Economic Betting" rather than rigid mathematical constraints. A critical feature of this framework is the rejection of hard-coded temporal parameters, recognizing that in a truly decentralized environment, static thresholds are both unenforceable and antithetical to network autonomy.

Within this framework, determining the 90% propagation threshold is not a matter of provable fact, but a rational, intelligent decision-making process performed by individual nodes. Each miner autonomously records the arrival timestamps of the $n - th$ full block and the subsequent compact blocks $(n + i, n + k)$. By calculating the observed arrival interval ΔT, a miner independently assesses the probability of global availability. If ΔT significantly exceeds the known network diameter, the miner "bets" on the truth—not out of blind obedience to a protocol, but as a strategic maneuver to avoid the certain loss of block rewards associated with opposing a better-informed majority.

The protocol shifts the burden of data availability from the consensus rules to the economic actors. If the $n - th$ full block is excluded from the chain due to a "False" declaration (a B-type fork), this is not a failure of the consensus mechanism, but a natural stochastic convergence of the network. Under the fluid nature of Nakamoto Consensus, the chain inevitably flows toward the direction supported by the majority hash power. Consequently, the onus of persistence lies with the original miner of the $n - th$ block; to avoid exclusion, the miner must disseminate the full block data with maximum urgency immediately following the i −confirmation gate. Any failure to reach the threshold before the arrival of the $(n + k) - th$ block is treated as a standard orphan event, reinforcing the principle that propagation efficiency is a prerequisite for economic finality.

A rational $(n + k) - th$ miner recognizes the fundamental asymmetry in success probabilities: a "True" declaration for a widely disseminated block yields a convergence probability of nearly 1, ensuring immediate acceptance into the canonical chain. Conversely, a "False" declaration in a state of high data availability introduces complex, non-deterministic risks that

threaten the miner's own ROI. This economic gravity ensures that, under normal network assumptions and in the absence of intentional withholding, the $n-th$ block will achieve high-probability finality before its evaluation at height $n+k$.

Ultimately, the robustness of the system does not rely on hard-coding ΔT, but on the alignment of individual greed with global truth. Even when valid blocks are excluded due to propagation delays, the system converges according to predefined stochastic rules; this is, in essence, a process of objectively reflecting the current state of the network. By transforming "propagation uncertainty" into a "competitive engine," the $n_k_declaration$ effectively decouples transaction throughput from physical latency, ensuring that the Nash Equilibrium consistently converges with actual data availability through the natural decentralized consensus.

Consequently, the following operational requirements are identified for system integrity:

Parameter k Calibration: The value of k must be configured sufficiently large to allow the $n-th$ full block to be fully disseminated across the network, ensuring that the majority of hashing power can validate the block before the arrival of the $n+k-th$ declaration.

Mandatory Rapid Propagation: Mining nodes that release a $n-th$ compact block must propagate the corresponding full block data as expeditiously as possible to the entire network; any delay in data availability may be misinterpreted as a malicious false declaration, leading to the deliberate orphaning of the block by rational peers.

Requirement of Private Chain Domination: After mining the $n+k-th$ block with a “False” declaration, the adversary is effectively isolated. Since honest nodes will not build upon that chain, the attacker must continue extending the private branch alone. To successfully override the honest chain, the adversary must mine a longer chain $n+k+j(j>0)$ before the honest majority extends their branch.

This implies:

- No propagation advantages
- No honest hash redirection
- Pure private PoW race

Reduction to conditional private fork attack: Since no honest miner contributes to the attacker’s chain under “False” declaration, the adversary must independently accumulate sufficient PoW to surpass the honest chain. Unlike Bitcoin, where winning a single block race at height $n+k$ can immediately unify the network, the proposed protocol structurally prevents such unification under declaration inconsistency.

Forgery is a strictly dominated conditional private fork attack that yields no strategic advantage over original private fork attack as follows.

After block $n-1$ is published, the adversary begins mining a private branch in secret. The attack is declared successful only if the adversary achieves a lead of at least one block strictly after the mining event corresponding to block n. Thus, even if the adversary mines block n immediately after publication, this event alone does not constitute success.

This additional constraint modifies the stochastic structure of fork competition in a mathematically nontrivial way.

Let

$$\alpha \in (0,1/2)$$

denote the adversary’s relative hash power, and define

$$\beta = 1-\alpha$$

as the relative hash power of the honest network.

Define the state variable

$$\Delta_t = a_t - h_t,$$

where:

a_t is the length of the adversary’s private chain extension beyond block $n-1$,

h_t is the length of the honest network’s public chain extension beyond block $n-1$.

The process evolves as a discrete-time biased random walk:

$$\Delta_{t+1} = \begin{cases} \Delta_t + 1, & \text{with probability } \alpha \\ \Delta_t - 1, & \text{with probability } \beta \end{cases}$$

with initial condition

$$\Delta_0 = 0.$$

The attack succeeds if

$$\exists\, t \geq 2 \text{ such that } \Delta_t \geq 1.$$

Importantly,

$$\Delta_t = 1$$

is excluded from the success set, since it corresponds to the adversary mining block n immediately after the publication of block $n-1$.

Consistent with Theorems 1 and 2, the difficulty penalty η must be bound strictly to the adversary's first private block allocation ($k_{adv} = 1$). The initial state-dependent transition probabilities under the penalty regime are defined as:

$$\alpha^* = \frac{\frac{\alpha}{\eta}}{\frac{\alpha}{\eta} + (1-\alpha)}, \qquad \beta^* = \frac{1-\alpha}{\frac{\alpha}{\eta} + (1-\alpha)}$$

The state space behaves as an inhomogeneous random walk governed by whether the adversary has successfully cleared their penalty. The system branches into two disjoint spaces based on the first mining event:

Theorem 3 (Exact Success Probability).

Under the proposed block-indexed penalty model, the exact probability of attack success for $\alpha < 1/2$ is given by:

$$P_{success} = \frac{\alpha^2(2\alpha + \eta(3-4\alpha))}{(\alpha + \eta(1-\alpha))(\alpha + \eta(1-2\alpha))}$$

Proof. We condition on the first mining event after the publication of block $n-1$.

Case 1: The adversary mines first (probability α^*)

The state transitions to $\Delta_1 = 1$. Since the adversary has successfully mined their first block, the penalty is cleared ($k_{adv} = 1$). All subsequent steps resume at the base hash power (α, β). We evaluate two subcases from $\Delta_1 = 1$:

1. The adversary mines again (probability α):

   $$\Delta_1 = 1 \rightarrow \Delta_2 = 2$$

   This yields immediate success. Contribution : $\alpha^* \cdot \alpha$

2. The honest network mines (probability β):

   $$\Delta_1 = 1 \rightarrow \Delta_2 = 0$$

   From this neutralized state $\Delta = 0$, the penalty is already spent. The probability of a baseline random walk ever reaching the absorbing boundary +1 is given by the classical Gambler's Ruin probability α/β. Contribution : $\alpha^* \cdot \beta \cdot \alpha/\beta = \alpha^* \cdot \alpha$.

   Summing the branches, the total contribution from Case 1 is:

$$Cont_{case1} = 2\alpha^*\alpha = \frac{2\frac{\alpha}{\eta}\alpha}{\frac{\alpha}{\eta} + (1-\alpha)} = \frac{2\alpha^2}{\alpha + \eta(1-\alpha)}$$

Case 2: The honest network mines first (probability β^*)

The state transitions to $\Delta_1 = -1$, and the adversary still holds zero private blocks ($k_{adv} = 0$). Consequently, the penalty η remains active and will trap the adversary on their very first successful mining step, regardless of how deep the honest network extends the chain.

Let $j \geq 0$ be the exact number of additional blocks mined by the honest network before the adversary finally forces their first private block. The probability of the honest network mining exactly j consecutive blocks followed by 1 adversary block within the penalty regime is:

$$P(First\ Private\ Block\ at\ depth\ j) = (\beta^*)^j \cdot \alpha^*$$

At the exact moment this block is mined, the penalty is cleared, and the state becomes:

$$\Delta = -1 - j + 1 = -j$$

Since the penalty is now eliminated, the process transforms into a standard homogeneous random walk starting from state $-j$. The probability of ever climbing up to the success threshold $+1$ from a deficit of $-j$ steps is:

$$P(Reach\ +1\ from\ -j) = (\frac{\alpha}{\beta})^{j+1} = (\frac{\alpha}{1-\alpha})^{j+1}$$

To find the true total contribution of Case 2, we integrate over all possible public depths j via an infinite geometric series:

$$Cont_{case2} = \beta^* \sum_{j=0}^{\infty} [(\beta^*)^j \alpha^* \cdot (\frac{\alpha}{1-\alpha})^{j+1}]$$

Factoring out the terms independent of j:

$$Cont_{case2} = \beta^* \cdot \alpha^* (\frac{\alpha}{1-\alpha}) \sum_{j=0}^{\infty} (\beta^* \frac{\alpha}{1-\alpha})^j$$

Let the common ratio be

$$R = \beta^* \frac{\alpha}{1-\alpha}.$$

Substituting

$$\beta^* = \frac{\eta(1-\alpha)}{\alpha + \eta(1-\alpha)}$$

yields the following algebraic simplification:

$$R = \frac{\eta(1-\alpha)}{\alpha + \eta(1-\alpha)} \cdot \frac{\alpha}{1-\alpha} = \frac{\eta\alpha}{\alpha + \eta(1-\alpha)}$$

Using the geometric series summation formula $\frac{1}{1-R}$:

$$\frac{1}{1-R} = \frac{1}{1 - \frac{\eta\alpha}{\alpha + \eta(1-\alpha)}} = \frac{\alpha + \eta(1-\alpha)}{\alpha + \eta(1-\alpha) - \eta\alpha} = \frac{\alpha + \eta(1-\alpha)}{\alpha + \eta(1-2\alpha)}$$

Now, substitute this sum back into the contribution equation:

$$Cont_{case2} = \beta^* \cdot \alpha^* \cdot (\frac{\alpha}{1-\alpha}) \cdot \frac{\alpha + \eta(1-\alpha)}{\alpha + \eta(1-2\alpha)}$$

Notice that $\beta^* \cdot (\alpha + \eta(1-\alpha)) = \eta(1-\alpha)$. Substituting this eliminates the $(1-\alpha)$ term in the denominator:

$$Cont_{case2} = \alpha^* \cdot \frac{\eta\alpha}{\alpha + \eta(1-2\alpha)} = \frac{\eta\alpha^2}{(\alpha + \eta(1-\alpha))(\alpha + \eta(1-2\alpha))}$$

Summation

Combining $Cont_{case1}$ and $Cont_{case2}$ by finding the common denominator $(\alpha + \eta(1-\alpha))(\alpha + \eta(1-2\alpha))$:

$$P_{success} = \frac{2\alpha^2(\alpha + \eta(1-2\alpha)) + \eta\alpha^2}{(\alpha + \eta(1-\alpha))(\alpha + \eta(1-2\alpha))}.$$

Expanding the numerator:

$$\text{Numerator} = \alpha^2(2\alpha + 2\eta(1-2\alpha) + \eta) = \alpha^2(2\alpha + 2\eta - 4\eta\alpha + \eta) = \alpha^2(2\alpha + \eta(3-4\alpha))$$

Thus, we obtain the closed-form expression for the exact success probability:

$$P_{success} = \frac{\alpha^2(2\alpha + \eta(3-4\alpha))}{(\alpha + \eta(1-\alpha))(\alpha + \eta(1-2\alpha))} < \alpha/(1-\alpha). \quad \square$$

Consequently, "False" $n_k_declaration$ Attack constitutes a conditional private fork attack, which yields a lower success rate than other well-known attacks on Bitcoin, such as the original private fork attack or selfish mining.

4.2.3. Economic Unprofitability of Data Withholding under True n_k_declarations

Section 4.2.2 examined attacks in which an adversary sets $n_k_declaration$ to False. This section addresses the complementary case: attacks based on a True declaration. A True declaration for a non-existent Full Block constitutes a ghost-chain attack. Under the proposed protocol, such an attack is immediately frustrated because honest nodes cannot verify the declared block and therefore reject the declaration at the consensus boundary. The only remaining True-declaration attack is therefore the data withholding attack, in which the $n-th$ Full Block exists and is valid, but its propagation is deliberately delayed or withheld while the attacker declares True at height $n+k$.

The only party capable of withholding the $n-th$ Full Block is the group that mined block nn, whose hash power is denoted by α. The mining of intermediate blocks $n+1, \ldots, n+k-1$ is irrelevant to this attack; regardless of which party mines them, the fate of block nn is determined solely by who first mines block $n+k$.

Theorem 4 (No Profitable Data Withholding under True Declaration).
Under the proposed protocol with penalty coefficient $\eta > 1$, a data withholding attack based on a True declaration yields an expected reward strictly lower than honest mining whenever $\alpha < 1/(\eta - 1)$. In particular, for $\eta = 1.2$, profitability would require $\alpha > 5$, which is impossible. Hence no rational selfish miner has an incentive to mount this attack.

Proof. Consider first the honest strategy. If the miner of block nn propagates the Full Block normally and continues honest mining, R_n is guaranteed. The expected reward from block $n+k$ is αR_{n+k} , since the miner controls fraction α of the hash power. Hence

$$E_{normal} = R_n + \alpha R_{n+k}.$$

Assuming $R_n = R_{n+k} = R$,

$$E_{normal} = (1+\alpha)R.$$

Now consider the data withholding attack. The attacker withholds the $n-th$ Full Block, causing honest nodes to lack the data required to declare block n valid at height $n+k$. Honest nodes therefore produce a block with $n_k_declaration = False$ and are subject to the difficulty penalty $\eta = 1.2$. Their effective hash power is thus $(1-\alpha)/\eta$, whereas the attacker mines with hash power α at the normal difficulty under a True declaration. The probability that the attacker wins the race at height $n+k$ is therefore

$$P_{win} = \alpha/(\alpha + (1-\alpha)/\eta).$$

If the attacker wins, block n is finalized as True and the attacker also obtains block $n+k$; the reward is $R_n + R_{n+k}$. If the attacker loses, the honest chain declares block n as False, so block n is invalidated and the attacker receives neither R_n nor R_{n+k}; the reward is 0. Thus

$$E_{attack} = P_{win}(R_n + R_{n+k}) + (1 - P_{win}) \cdot 0 = P_{win}(R_n + R_{n+k}).$$

Assuming $R_n = R_{n+k} = R$,

$$E_{attack} = 2P_{win}R.$$

The attack is profitable only if $E_{attack} > E_{normal}$. Hence

$$P_{win}(R_n + R_{n+k}) > R_n + \alpha R_{n+k}.$$

With $R_n = R_{n+k} = R$, this becomes

$$2P_{win}R > (1+\alpha)R,$$

or equivalently,

$$2P_{win} > 1 + \alpha.$$

Substituting P_{win},

$$2\alpha/(\alpha + (1-\alpha)/\eta) > 1 + \alpha.$$

Multiplying both sides by $\alpha + (1-\alpha)/\eta > 0$ and simplifying yields

$$\alpha(1-\alpha) > (1-\alpha)(1+\alpha)/\eta.$$

Since $\alpha < 1$, dividing by $1 - \alpha > 0$ gives

$$\alpha > (1+\alpha)/\eta,$$

or equivalently,

$$(\eta - 1)\alpha > 1, \text{ i.e. } \alpha > 1/(\eta - 1).$$

For $\eta = 1.2$, this requires

$$\alpha > 1/0.2 = 5,$$

which is impossible for any realistic hash-power fraction. Therefore, for all realistic $\alpha < 1$, the expected reward from the data withholding attack is strictly lower than that from honest mining. A rational selfish miner will not launch this attack. □

5. PERFORMANCE ANALYSIS

The probabilistic security rationale and the correspondence between Bitcoin's propagation-delay assumption and the present k −block declaration window have already been established in the Remark following Theorem 1 and are therefore not repeated here. For the performance evaluation, the relevant question is whether the selected parameters—10 MB Full Blocks, a 600-second block interval, $k = 6$ and $i = 2$—remain within the safe operating region identified above. These values are chosen conservatively so that P, the probability that all honest nodes possess and have verified the Full Block B_n before the $(n + k) - th$ block is mined, remains sufficiently high. The following subsections analyze Compact Block capacity, propagation latency, parameter validation, and throughput scaling under this configuration.

5.1. Compact Block Capacity and Efficiency Analysis

The efficiency of the proposed Compact Block is governed by the bit-length of compressed UTXO IDs and the resulting probability of collisions. While conventional Compact Block Relay (CBR) suffers from critical propagation stalls when a collision occurs requiring complex request response cycles to resolve the ambiguity, our protocol remains robust. In our system, a collision merely results in "over-freezing," where a few additional UTXOs are temporarily locked, posing no threat to the consensus continuity or propagation speed.

5.1.1. Mathematical Analysis of Collision Probability and Over Freezing

To determine the optimal ID size, we analyze the collision expectations based on a 10MB block size.

System Parameters: A 10MB block contains approximately 40,000 transactions. With an average of 2.5 inputs per transaction, the number of UTXOs to be compressed per block is 100,000.

Mempool Scale: Assuming a mempool of 500,000 pending transactions, the total number of UTXO IDs monitored is 1,250,000.

Collision Expectation: The expected number of collisions in a set of n elements with an ID space of $H = 2^b$ is approximately $E = n^2/(2H)$. Considering that a collision involves at least two UTXOs, the expected count of UTXOs participating in collisions is approximately $2 \times E$. We analyze ID sizes of 4, 5, 6, 7 and 8 bytes. To maintain a negligible over-freezing rate even in high throughput environments, a 6-byte (48-bit) ID is selected as the optimal balance between compression and collision resistance (Table 5). As presented in Table 5, when using 6-byte compressed hashes, the expected number of UTXOs involved in hash collisions within the mempool is less than 1. Even in the rare event of a collision, "over-freezing" only affects an insignificant fraction of UTXOs in the mempool, ensuring that honest transactions are virtually unaffected.

Table 5: Collision Expectation

ID Size (Bytes)	Bit-length (b)	ID Space (H)	Expected UTXOs in Collision ($n = 1.25 \times 10^6$)
4 Bytes	32-bit	4.29×10^9	364
5 Bytes	40-bit	1.10×10^{12}	1.42
6 Bytes	48-bit	2.81×10^{14}	0.0055
7 Bytes	56-bit	7.20×10^{16}	0.0000217
8 Bytes	64-bit	1.84×10^{19}	0.0000000847

5.1.2. Comparison with Conventional CBR

Unlike conventional CBR which compresses entire transactions, our protocol compresses individual UTXOs used as inputs.

ID Count: Historical Bitcoin data shows an average of 2.5 inputs per transaction. Thus, our Compact Block contains approximately 2.5 times more IDs than CBR.

Propagation Superiority: Although the data volume is slightly higher than CBR, our Compact Block achieves faster effective propagation. Conventional CBR often stalls due to missing transactions in the local mempool, requiring "Request-Response" cycles. By utilizing a list of UTXO IDs for immediate validation, our Compact Block eliminates round trips and achieves performance equivalent to having a 100% synchronized mempool.

5.1.3. Independence from Mempool Synchronization

The proposed protocol fundamentally decouples block propagation from mempool state. Upon receiving a valid Compact Block, a node can immediately proceed to mine the next block. This is because the block's validity for mining purposes is determined solely by the verification of its header (Proof-of-Work) and the objective consistency of its $n_k_declaration$. The mempool is not consulted for transaction reconstruction at this stage. This design choice significantly mitigates the system's sensitivity to mempool asynchrony, a critical weakness in conventional CBR. However, an indirect, second order influence exists on network wide efficiency. The subsequent, Full Block propagation phase (which is decoupled from the mining cycle) can leverage fast relay protocols like CBR [25-29]. The efficiency of this Full Block propagation is, as in any system, dependent on mempool synchronization. Poor mempool synchronization can slow the diffusion of the 10MB Full Block data across the network. This delay only affects the background data synchronization. In our protocol, the mempool's role is demoted from a critical path dependency to a mere background efficiency optimizer.

5.2. Network Propagation Simulation, Parameter Validation, and Throughput Scaling

This section provides a comparative simulation of propagation delays and validates the security parameter k under scaled block sizes, ultimately demonstrating the protocol's ability to increase transaction throughput without compromising security. Specifically, to ensure the robustness of our analysis, the values used in this simulation were conservatively assumed with an integrated safety margin, rendering them more restrictive than typical real-world data. Additionally, these timings serve only as a relative guideline rather than absolute values, given the inherently dynamic and unpredictable nature of network states.

5.2.1. Compact Block Propagation Delay Simulation and Comparison

To ground our analysis in realistic network dynamics, we model propagation delays based on established Bitcoin network research [30-36] and the fundamental differences in protocol design.

A. Proposed Protocol: Compact Block Propagation

By scale-adjusting our model, we evaluate the performance of Compact Blocks across various Full Block sizes. By employing a 6-byte compressed UTXO ID for each input, the size of our Compact Block remains fixed relative to the number of inputs. For instance, a Compact Block corresponding to a 10MB Full Block (containing 100,000 inputs) is a fixed 600 KB

message. The propagation time (δ_{CB}) of the Compact Block is dominated by network latency and bandwidth. Based on models for Bitcoin block propagation, the time required for a Compact Block to reach 90% of nodes in a well-connected Bitcoin-like network is simulated as follows.

In our scheme, Compact Blocks do not require mempool-dependent recovery, ensuring that the propagation delay remains highly stable and independent of the mempool state.

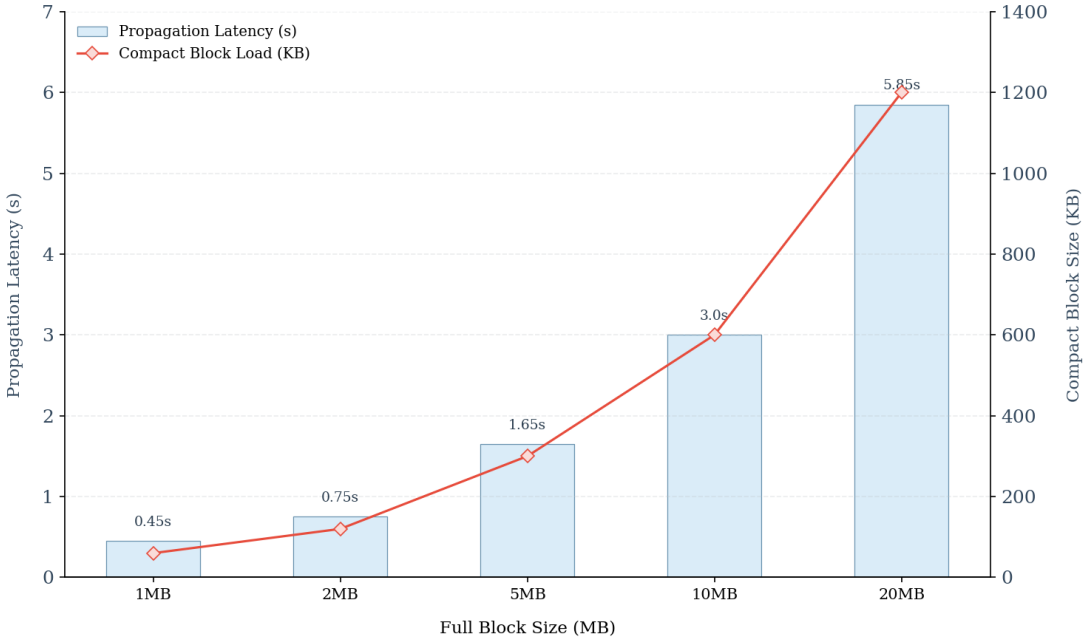


Figure 5. Performance Analysis: Propagation Delay with Compact Block Scaling

B. Bitcoin with CBR: 1 MB Block Propagation

The propagation of a 1 MB block using CBR is a two-phase stochastic process: transmission of a small Compact Block (~50 KB), and recovery of missing transactions via separate requests. The total time (δ_{BTC}) is heavily dependent on mempool synchronization between the relaying and receiving nodes. We simulate δ_{BTC} for the block to reach 90% of nodes under varying mempool sync rates, drawing from empirical studies of CBR performance.

Table 6: Simulated 90% Network Propagation Delay for a 1 MB Block (Bitcoin with CBR)

Mempool Synchronization Rate	Simulated Propagation Delay (δ_{BTC})	Primary Bottleneck
70%	~12s	Regular recovery cycles
90%	~5s	Occasional recovery
95%	~3s	Near optimal CBR performance

C. Comparison and Implication

Consistent with the Bitcoin network, the block finding process in our scheme is modeled as a Poisson process with a rate of $\lambda = 1/600s^{-1}$. Under this model, the probability of a fork occurring during the Compact Block propagation delay δ_{CB} is given by

$$P_{CB} = 1 - e^{-\lambda\delta_{CB}} \approx \lambda\delta_{CB}.$$

In contrast, the fork probability for a standard Bitcoin block is expressed as

$$P_{BTC} = 1 - e^{-\lambda\delta_{BTC}} \approx \lambda\delta_{BTC}.$$

From this, it can be inferred that if $\delta_{CB} < \delta_{BTC}$, then $P_{CB} < P_{BTC}$. While the peak performance of optimal CBR (~3 seconds at 95% sync) is comparable to our Compact Block's propagation (~3 seconds) for a 10MB Full Block (Figure 5), our protocol eliminates the performance variance and the risk of extreme delays associated with low mempool sync rates.

The proposed system delivers consistently low latency propagation, effectively decoupling the critical path for consensus from the unpredictable state of the mempool. This results in an effective network hesitation period (δ_{CB}) that is as good as or better than Bitcoin's under optimal conditions (δ_{BTC}) and significantly superior under non-ideal, real-world conditions. In our scheme, the Full Block size can be configured up to a maximum of 10MB, as the condition $\delta_{CB} \leq \delta_{BTC}$ is no longer satisfied once the block size exceeds this threshold (Figure 5). It should be noted, however, that this is based on a considerably conservative set of assumptions.

Since each block discovery is an independent event (memoryless property of the Poisson process), the probability of a fork extending to a specific depth k is roughly the probability of k consecutive "near-simultaneous" mining events occurring before the network synchronizes.

The probability of a fork reaching depth k is:

$$P(depth \geq k) \approx P_{BTC}^{k}$$

Assuming a propagation delay of $\delta_{BTC} = 3$ seconds, which yields a single block fork probability of $P_{BTC} = 3/600 = 0.005$, the probability of a fork reaching a depth of $k \geq 2$ is calculated as $P(depth \geq 2) \approx (0.005)^2 = 0.000025$ (or 1 in 40,000 blocks). This represents a statistically negligible probability, implying that at a standard generation rate of 144 blocks per day, such an event is expected to occur approximately once every nine months. From this, we establish $i = 2$ as the default confirmation threshold to ensure general network stability.

5.2.2. Full Block Propagation and Security Analysis of Parameter r

A. Full Block Propagation Simulation

The 10 MB Full Block propagation can utilize fast relay protocols like CBR. We simulate its 90% propagation time (δ_{FB}) under the same mempool sync conditions, scaling the delays from the 1 MB model to account for the larger data volume and longer recovery phases.

Table 7: Simulated 90% Network Propagation Delay for 10 MB Full Block

Mempool Synchronization Rate	Simulated Propagation Delay (δ_{FB})	Primary Bottleneck
70%	~90s	Regular recovery cycles
90%	~35s	Occasional recovery
95%	~20s	Near optimal CBR performance

B. Probabilistic Stability and Nested Fork Resilience

To establish a rigorous security boundary for the proposed pipelined consensus, we evaluate the stochastic stability of the protocol under a normal network assumption. The security of the asynchronous structure relies on the $n_k_declaration$ mechanism, subject to the i-confirmation gated propagation rule. We model the mining process as a Poisson process with intensity $\lambda = 1/600s^{-1}$ and a conservative Full Block propagation delay $\delta_{FB} = 35s$ for 10MB Full Blocks.

We define four critical stochastic events to quantify the resilience of the protocol.

Event P_1 (Initial Declaration Fork):

P_1 occurs when the r blocks (where $r = k - i$) following the gate-triggering block are mined before the $n - th$ Full Block is disseminated to over 90% of the network. For $i = 2$, this means the sequence from $(n + i + 1)$ to $(n + k)$ is completed within δ_{FB}. This causes a divergence in the $n_k_declaration$ at height $n + k$, splitting nodes into a “True” group (Group A) and a “False” group (Group B).

The number of blocks $N(t)$ generated within time interval t follows a Poisson distribution with parameter λt:

$$N(t) \sim Poisson(\lambda t)$$

The probability of exactly r blocks being mined within time t is:

$$P(N(t) = r) = \frac{e^{-\lambda t}(\lambda t)^r}{r!}, r = 0,1,2, \ldots$$

The probability of mining at least r blocks within time t is the complement of the probability of mining fewer than r blocks:

$$P(N(t) \geq r) = \sum_{j=r}^{\infty} \frac{e^{-\lambda t}(\lambda t)^j}{j!} = 1 - \sum_{j=0}^{r-1} \frac{e^{-\lambda t}(\lambda t)^j}{j!} = 1 - e^{-\lambda t} \sum_{j=0}^{r-1} \frac{(\lambda t)^j}{j!}$$

Event P_2 (Sequential Fork):

P_2 occurs when the r blocks are mined before the $(n+1) - th$ Full Block completes its dissemination.

Then, P_1 and P_2 are denoted by

$$P_1 = P_2 = P(N(\delta_{FB}) \geq r).$$

Table 8 presents the upper bounds of δ_{FB} satisfying $P_1 < \boldsymbol{\varepsilon}$ corresponding to different values of r.

Table 8: Critical Time Values (δ_{FB}) satisfying ($P_1 < \boldsymbol{\varepsilon}$)

r	$\boldsymbol{\varepsilon} = 0.001$	$\boldsymbol{\varepsilon} = 0.0001$
2	$\delta_{FB} < 27.24$s	$\delta_{FB} < 8.52$s
3	$\delta_{FB} < 114.32$s	$\delta_{FB} < 51.7$s
4	$\delta_{FB} < 257.13$s	$\delta_{FB} < 139.07$s
5	$\delta_{FB} < 443.62$s	$\delta_{FB} < 266.67$s
6	$\delta_{FB} < 664.26$s	$\delta_{FB} < 428.24$s

Event P_3 (Nested Fork):

P_3 represents the simultaneous occurrence of P_1 and a sequential fork P_2. Under the i-confirmation rule, the propagation windows for the $n - th$ and $(n+1) - th$ blocks are triggered by the $(n+i) - th$ and $(n+i+1) - th$ blocks, respectively. Consequently, P_1 and P_2 are inherently dependent as their governing mining intervals overlap significantly across the sliding window.

To model this, we consider the inter-arrival time of blocks. Let y be the cumulative time required to mine the first $(r-1)$ blocks within the interval ([0, δ_{FB}]). The probability density function (PDF) for y follows the Erlang distribution:

$$f(y; r-1, \lambda) = \frac{\lambda^{r-1} y^{r-2} e^{-\lambda y}}{(r-2)!}$$

For a nested fork to occur, at least one additional block must be mined within the remaining interval $(\delta_{FB} - y)$ for both the current and the subsequent dissemination windows. By integrating this joint condition over the dissemination period ([0, δ_{FB}]), we obtain:

$$P_3 = \int_0^{\delta_{FB}} \frac{\lambda^{r-1} y^{r-2} e^{-\lambda y}}{(r-2)!} \cdot (1 - e^{-\lambda(\delta_{FB}-y)})^2 dy$$

Event P_4 (Cumulative Delay Fork):

P_4 is an extreme case where $r+1$ blocks are discovered within a single dissemination window δ_{FB}. This represents a higher-order mining burst compared to P_4:

$$P_4 = P(N(\delta_{FB}) \geq r+1) = 1 - \sum_{j=0}^{r} \frac{e^{-\lambda\delta_{FB}} (\lambda\delta_{FB})^j}{j!}$$

P_4 is formally defined as the probability that the $n-th$ Full Block, which triggered an initial declaration fork at the $(n+k)-th$ block, fails to reach more than 90% of the network nodes before the $(n+k+1)-th$ block is mined. A negligible value of P_4 signifies that data availability (DA) is robustly guaranteed at the critical juncture of fork resolution. This ensures that the majority of nodes possess the necessary block data to identify the canonical chain, thereby minimizing the risk of persistent divergence and ensuring a timely transition (Section 4.1.2.C) to a unique consensus state.

Applying the parameters ($\delta_{FB} = 35s, \lambda = 1/600, i = 2$), we evaluate the probabilities across varying pipeline depths r(Table 9).

Table 9: Probabilistic Stability Assessment ($\boldsymbol{\delta_{FB} = 35s, \lambda = 1/600, i = 2}$)

$r(k)$	P_1(Single)	P_3(Nested)	P_4(Cumulative)
2(4)	1.64×10^{-3}	6.24×10^{-5}	3.12×10^{-5}
3(5)	3.12×10^{-5}	9.1×10^{-7}	4.56×10^{-7}
4(6)	4.56×10^{-7}	1.06×10^{-8}	5.31×10^{-9}
5(7)	5.31×10^{-9}	1.03×10^{-10}	5.19×10^{-11}
6(8)	5.19×10^{-11}	8.6×10^{-13}	4.31×10^{-13}

C. Validation of r and Throughput-Security Proposition

The simulation and probability analysis lead to two key conclusions:

Parameter Security: As shown in Table 8 and 9, the upper bound of the propagation delay that maintains a negligible probability of a $n_k_declaration$ fork is maximized when $r = 6$. This validates $r = 6$ as a highly conservative and secure choice. While $r = 6$ offers an immense mathematical safety margin, our analysis demonstrates that $r = 4$ is already functionally secure, with the $P_1 < 4.56 \times 10^{-7}$, $P_3 < 1.06 \times 10^{-8}$ and $P_4 < 5.31 \times 10^{-9}$.

The proof of Theorem 1 in this study is predicated on the following implicit assumptions, which align with the classical convergence models of Bitcoin and other Nakamoto-style consensus protocols:

- **Assumption 1 (High Propagation Reliability):** It is assumed that once an initial fork occurs, the full block causing the fork reaches the vast majority of nodes with a probability approaching unity before the discovery of the subsequent block.
- **Assumption 2 (Negligibility of Nested Forks):** The probability of nested forks (P_3) is assumed to be sufficiently low to be negligible in a practical operational environment.

Under these assumptions, a chain length differential of even a single block (depth 1) guarantees facilitates highly probable convergence within a single-block differential under standard operational assumptions. This logic is consistent with Bitcoin's

security model, which assumes that blocks are disseminated rapidly enough relative to the mining interval that simultaneous discoveries are rare and short-lived. In this regard, the stability proof of the proposed protocol maintains theoretical continuity with Bitcoin's foundational principles.

Even in the extreme event of a nested fork or prolonged network partition, the system is designed to achieve unique convergence through PoW Gravity (Section 3.4.2.B), which pulls the consensus towards the chain with the greatest cumulative work. However, high frequencies of nested forks can undermine system liveness and utility. Therefore, we have conservatively calibrated the system parameters ($r = 4, i = 2, k = 6$) to ensure that Assumptions 1 and 2 are robustly satisfied, mirroring the conservative parameterization found in the Bitcoin network. This choice strikes an optimal balance between maintaining a robust security posture and enhancing the responsiveness of the network. Nevertheless, as a more conservative alternative for enhanced safety, the system can also be configured with an 8-byte UTXO compressed ID and $r = 6(i = 3, k = 9)$. In this configuration, the network achieves an even higher level of cryptographic security and fork resistance.

Throughput Scaling without Security Loss: The core result is that increasing the block size by a factor of 10 (from 1 MB to 10 MB) to increase transaction throughput does not necessitate an increase in the security parameter r. Specifically, the $k = 6$ ($i = 2, r = 4$) configuration is designed to offer a transaction finality time equivalent to the traditional 6-confirmation standard in Bitcoin. The structural decoupling of consensus ($n_k_declaration$ via Compact Block) from data availability (Full Block propagation) allows the system to scale data capacity while maintaining the same, or even improved, resilience against propagation-triggered forks compared to the base Bitcoin network ($P_{CB} \leq P_{BTC}, P_{FB} \leq P_{BTC}$). The security of the consensus layer remains intact and independent of the scaling of the data layer.

The security analysis of Bitcoin is predicated on the implicit assumption that the block propagation delay is significantly shorter than the block mining interval. Similarly, the security arguments presented in Section 4 for our proposed scheme are based on the following implicit assumptions:

- The propagation delay of a Full Block is significantly shorter than the time required to mine r blocks.
- The propagation time of a Compact Block corresponding to a single block is shorter than that of a standard 1MB Full Block in the Bitcoin network.

These assumptions are fundamentally guaranteed by the parameter configurations established above (10MB block size, 600-second block interval, $k = 6$ and $i = 2$).

Finally, it should be noted that in cases of long-term network partitions, the reorganization (reorg) cost incurred during the convergence process is a function of the partition duration and the system's throughput (TPS). This cost is not a flaw unique to our proposed asynchronous pipeline but is an inherent characteristic shared by all PoW-based consensus mechanisms governed by the CAP theorem.

5.2.3. Throughput Calculation and Scaling

The selection of system parameters is critical for balancing security and throughput. For simulation and safety margin maximization, we established $k = 6$ (evaluation window), $i = 2$ (Full Block propagation gate), and a 10MB Block Size while maintaining the Bitcoin-standard 600-second interval. Assuming an average transaction size of 250 bytes, a 10MB block accommodates approximately 40,000 transactions. The transaction throughput (TPS) is calculated as follows:

$$Throughput(TPS) = \frac{10{,}000{,}000\ bytes / 250\ bytes\ per\ TX}{600\ seconds} = \frac{40{,}000}{600} \approx 66.67\ TPS$$

This represents a nearly ten-fold increase compared to the standard Bitcoin protocol (~7 TPS) without requiring modifications to the mining frequency.

6. DISCUSSION

While the proposed framework introduces an asynchronous pipeline approach, it is important to acknowledge two aspects that could theoretically affect operational efficiency:

- **Impact of the k −Parameter on UTXO Expenditure Frequency:** In the exemplary schema used in our simulation, the parameter is configured as $k = 6$, thereby strictly prohibiting the expenditure of a UTXO within six blocks of its creation. Given that six blocks represent the standard threshold for transaction finality in Bitcoin, such a constraint aligns

with typical user behavior under conventional operating conditions. However, because UTXO expenditures are restricted for this duration, this constraint may impose a performance bottleneck in high-frequency service environments or decentralized finance (DeFi) protocols that require rapid, continuous capital turnover. Addressing this potential limitation remains a compelling research direction. A concrete resolution is proposed in Appendix B, where the Dual-Coinbase mechanism removes the mandatory inter-block freezing and liberates the parameter k from this economic constraint.

- **Efficiency in Scenarios with High Input-to-Transaction Ratios:** The efficiency of the proposed schema may decrease in hypothetical scenarios where the number of UTXO inputs per block significantly outweighs the total number of transactions. However, the probability of such occurrences is negligible in practice. For instance, in Bitcoin, the vast majority of transactions utilize only one or two inputs. This trend reflects not only a cost optimization strategy to minimize transaction fees but also a user preference for simplicity in asset management, aligning with the principle of cognitive economy.

In addition, it should be emphasized that the present study considers only the continuous superposition of pipeline stages. As an alternative, a hybrid design could be proposed in which, after a large block is mined, several subsequent standard-size blocks are mined according to the classical Bitcoin serial-verification procedure, so that pipelining is applied selectively to periodic large blocks rather than continuously. Because the declaration event in such a cadence recurs only once per large-block cycle and is therefore separated by multiple standard-size blocks, nested B-type forks are structurally suppressed, and Assumption 2 of the security analysis can therefore be dispensed with under normal network conditions. Moreover, since the freezing constraint is confined to the periodic large block rather than applied to every block, the average inventory of frozen UTXOs would be reduced substantially relative to the continuous-pipeline baseline. Such a hybrid cadence is not analyzed in this work and is left as a direction for future research.

The proposed mechanism for Compact Blocks and asynchronous consensus primarily focuses on enhancing network throughput. This approach offers the following scalability dimensions and design considerations:

- **Block Interval Reduction:** This proposal has primarily centered on the scenario of increasing the block size. However, the protocol's fundamental latency-hiding mechanism can be equally applied to reducing the block interval (consensus frequency). For example, in the alternative scenario where the block size is maintained at 1 MB while the interval is reduced to 100 seconds, transaction throughput can be increased while shortening the payment confirmation time to one-sixth of Bitcoin's baseline. Consequently, further granular evaluation is warranted to determine the optimal configurations balancing the mining interval, block size, and the values of k and i to maximize overall network efficiency (Appendix A).
- **Economic Incentive Realignment:** To induce the producer of the $(n + k) - th$ block to make a more responsible declaration regarding the validity of the $n - th$ block ($n_k_declaration$), an economic incentive mechanism could be introduced. A potential approach involves sharing a portion (e.g., 10%) of the $n - th$ block's reward with the $(n + k) - th$ block producer, contingent on the n-th block being finalized as valid. This mechanism provides a strong motivation for the $(n + k) - th$ producer to actively verify data availability and declare truthfully. The precise calibration of these reward ratios and their subsequent impact on network stability remains a promising avenue for further exploration.

7. CONCLUSION

This study proposed an asynchronous pipeline consensus framework designed to mitigate block propagation delays, a primary constraint in distributed networks, by decoupling block generation from verification dependencies. The technical analysis presented herein is not limited to a localized functional upgrade for a specific protocol like Bitcoin. Rather, it demonstrates potential for broader application across alternative blockchain systems. The structural features and contributions of this framework are summarized as follows:

- **Latency Hiding via Post-Declaration:** By separating block generation from finality confirmation, physical propagation latency is absorbed within subsequent computation intervals, thereby reducing operational idle time and potentially enhancing base-layer throughput.
- **Game-Theoretic Stability Realignment:** The protocol contributes to stability by integrating a work-weighted mechanism into the dynamic Difficulty Adjustment Algorithm. This approach protects the computational contributions of honest nodes and provides economic incentives to discourage inaccurate declarations.
- **Consensus-Agnostic Bottleneck Reduction:** As demonstrated in Appendix B, the Dual-Coinbase mechanism mitigates the economic constraints associated with temporary asset locking, allowing the pipeline parameter k to scale more

flexibly. Because this structural decomposition is inherently consensus-agnostic, it can be evaluated for various decentralized architectures—including PoW, PoS, BFT, and DAG—suggesting that the ultimate scalability limit remains fundamentally tied to the physical propagation velocity of the information-theoretic minimal consensus payload, the Compact Block.

- **Multi-Dimensional Parameter Optimization:** As explored in Appendix A, the framework offers a versatile design space for parameter optimization, extending beyond block size expansion to block interval reduction. This architectural flexibility allows networks to balance resource efficiency and fork resilience based on specific operational requirements.

In conclusion, the proposed protocol provides a practical pathway aimed at improving processing capability while maintaining compatibility with established distributed infrastructures, demonstrating its potential as a reliable data availability backbone for upper-layer scaling solutions.

REFERENCES

[1] K. Croman et al., "On Scaling Decentralized Blockchains," in Financial Cryptography and Data Security (LNCS 9604), pp. 106-125, 2016.
[2] Q. Zhou, H. Huang, Z. Zheng and J. Bian, "Solutions to Scalability of Blockchain: A Survey," in IEEE Access, vol. 8, pp. 16440-16455, 2020.
[3] M. J. M. Chowdhury, A. Colman, M. A. Kabir, J. Han and P. D. Sinderen, "A Survey on Blockchain Scalability," in Journal of Network and Computer Applications, vol. 127, pp. 35-58, 2019.
[4] A. Hafid, A. S. Hafid and M. Samih, "Scaling Blockchains: A Comprehensive Survey," in IEEE Access, vol. 8, pp. 125304-125320, 2020.
[5] T. A. Alghamdi, R. Khalid and N. Javaid, "A Survey of Blockchain Based Systems: Scalability Issues and Solutions, Applications and Future Challenges," in IEEE Access, vol. 12, pp. 79626-79651, 2024.
[6] G. A. F. Rebello, G. F. Camilo, L. A. C. de Souza, et al., "A Survey on Blockchain Scalability: From Hardware to Layer-Two Protocols," in IEEE Communications Surveys & Tutorials, vol. 26, no. 4, pp. 2411-2458, 2024.
[7] D. Khan, L. T. Jung and M. A. Hashmani, "Systematic Literature Review of Challenges in Blockchain Scalability," in Applied Sciences, vol. 11, no. 20, 9372, 2021.
[8] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," Technical Report, 2008. [Online]. Available: bitcoin.org
[9] J. A. Garay, A. Kiayias and N. Leonardos, "The Bitcoin Backbone Protocol: Analysis and Applications," in Advances in Cryptology – EUROCRYPT 2015, vol. 9057, pp. 281-313, 2015.
[10] I. Eyal, A. E. Gencer, E. G. Sirer and R. Van Renesse, "Bitcoin-NG: A Scalable Blockchain Protocol," in 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16), pp. 45-59, 2016.
[11] A. Sompolinsky and Y. Zohar, "Accelerating Bitcoin's Transaction Processing: BCH and GHOST," in International Conference on Financial Cryptography and Data Security (FC), pp. 37-53, 2015.
[12] A. Gangwal, H. R. Gangavalli and A. Thirupathi, "A Survey of Layer-two Blockchain Protocols," in Journal of Network and Computer Applications, vol. 209, p. 103539, Jan. 2023.
[13] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-chain Instant Payments," in Proc. Lightning Labs, San Francisco, CA, USA, vol. 9, Jan. 2016, p. 14.
[14] M. Corallo, "BIP 152: Compact Block Propagation," Bitcoin Improvement Proposals, 2016.
[15] A. P. Ozisik, G. Bissias and B. N. Levine, "Graphene: Efficient Interactive Set Reconciliation Applied to Blockchain Propagation," in ACM SIGCOMM Computer Communication Review, vol. 49, no. 1, pp. 16-27, 2019.
[16] M. Corallo, "The FIBRE Network," [Online]. Available: https://bitcoinfibre.org
[17] Y. Sompolinsky, Y. Lewenberg and A. Zohar, "SPECTRE: A Fast and Scalable Cryptocurrency Protocol," in Cryptology ePrint Archive, Report 2016/1159, 2016.
[18] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert and P. Saxena, "A Secure Sharding Protocol for Open Blockchains," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS), Vienna, Austria, Oct. 2016, pp. 17-30.
[19] M. B. Saif, S. Migliorini and F. Spoto, "A Survey on Data Availability in Layer 2 Blockchain Rollups: Open Challenges and Future Improvements," in Future Internet, vol. 16, no. 9, art. no. 315, Aug. 2024.
[20] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf and S. Capkun, "On the Security and Performance of Proof-of-Work Blockchains," in Proc. ACM CCS, pp. 3-16, 2016.
[21] X. Li, P. Jiang, T. Chen, X. Luo and Q. Wen, "A Survey on the Security of Blockchain Systems," in Future Generation Computer Systems, vol. 107, pp. 841–853, Jun. 2020.
[22] M. Saad et al., "Exploring the Attack Surface of Blockchain: A Comprehensive Survey," in IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1977-2008, 2020.
[23] I. Eyal and E. G. Sirer, "Majority is Not Enough: Bitcoin Mining is Vulnerable," in Proc. FC, pp. 436-454, 2014.
[24] A. Sapirshtein, Y. Sompolinsky and Z. Zohar, "Optimal Selfish Mining Strategies in Bitcoin," in Proc. FC, pp. 515-532, 2016.
[25] L. Zhang, T. Wang and S. C. Liew, "Speeding up block propagation in Bitcoin network: Uncoded and coded designs," in Computer Networks, vol. 206, art. no. 108791, Apr. 2022.
[26] J. Mišić, V. B. Mišić and X. Chang, "On the Benefits of Compact Blocks in Bitcoin," in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2020.
[27] C. Zhao, S. Zhang, T. Wang and S. C. Liew, "Bodyless block propagation: TPS fully scalable blockchain with pre-validation," in Future Generation Computer Systems, vol. 163, art. no. 107516, Feb. 2025.
[28] Z. H. Chin, V. M. Baskaran, C. K. Tan, I. K. T. Tan and T. T. V. Yap, "DAP-CBR: enhancing Bitcoin block propagation efficiency using dynamic compact block relay's prefilling of transactions," in The Journal of Supercomputing, vol. 81, art. no. 77, Jan. 2025.
[29] A. P. Ozisik, G. Andresen, G. Bissias, A. Houmansadr and B. Levine, "Graphene: A New Protocol for Block Propagation Using Set Reconciliation," in Data Privacy Management, Cryptocurrencies and Blockchain Technology, Cham, Switzerland: Springer, 2017, pp. 420–428.
[30] Y. Shahsavari, K. Zhang and C. Talhi, "A Theoretical Model for Block Propagation in Bitcoin-like Networks," in Proc. IEEE CNS, pp. 1-9, 2019.
[31] C. Decker and R. Wattenhofer, "Information Propagation in the Bitcoin Network," in Proc. IEEE P2P, pp. 1-10, 2013.
[32] M. Fadhil, G. Owenson and M. Adda, "A Bitcoin model for evaluation of clustering to improve propagation delay in Bitcoin network," in Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE), Aug. 2016, pp. 468–475.

[33] M. Fadhil, G. Owenson and M. Adda, "Locality based approach to improve propagation delay on the Bitcoin peer-to-peer network," in Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM), May 2017, pp. 556–559.
[34] M. Fadhil, G. Owenson and M. Adda, "Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network," in Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS), Jun. 2017, pp. 2411–2416.
[35] W. Bi, H. Yang and M. Zheng, "An accelerated method for message propagation in blockchain networks," 2018, arXiv:1809.00455. [Online]. Available: https://arxiv.org/abs/1809.00455
[36] Y. Aoki and K. Shudo, "Proximity neighbor selection in blockchain networks," 2019, arXiv:1906.00719. [Online]. Available: https://arxiv.org/abs/1906.00719

APPENDIX A. Architectural Comparison under Diverse Parameter Variations

The core contribution of this paper lies in the establishment of a novel blockchain architecture that decouples transaction validation dependencies through a pipelining mechanism, achieving substantial latency hiding. The specific parameters instantiated in the main text—such as the pipeline stages ($k = 6$), compressed ID size (6 bytes), block size (10 MB), and the 600 seconds mining interval—are mere exemplars utilized to demonstrate the theoretical feasibility of the framework. To prevent conflating rigid parameterizations with the underlying architecture, this appendix presents a comparative architectural analysis between the proposed framework and the traditional Bitcoin consensus model across varying parameter spaces.

A.1. The Role of Latency Hiding in Throughput Scaling

To establish a baseline, we assume an idealized Bitcoin network configuration: a block size of 1MB, a mining interval of 600s, and a network propagation threshold of 3s required for 90% of the nodes to receive a full block to suppress forks and guarantee consensus liveness.

A.1.1. Scenario A: Scaling Block Size to 10 MB (10 MB, 600s)

To determine the optimal ID size, we analyze the collision expectations based on a 10MB block size.

Bitcoin Architecture: In the traditional paradigm, expanding the block size to 10 MB causes an exponential surge in the fork rate. Because the time required to propagate a 10 MB block across 90% of the global network far exceeds the 3-second threshold, consensus degradation and block propagation bottlenecks become unavoidable.

Proposed Architecture: Under the proposed framework, while the physical network latency required for full block propagation remains identical to Bitcoin's baseline, the architectural deadline is expanded to 600 × 4 = 2,400 seconds. This marks an 800-fold increase in the permissible propagation window, which drastically mitigates the occurrence of B-Type forks compared to the Bitcoin baseline. To preserve real time consensus, the network requires only the high speed dissemination of compact blocks within a 3 second window. Unlike Bitcoin, this process eliminates block piping overheads and round trip retransmission delays caused by missing transactions, ensuring that A-Type fork rates remain statistically equivalent to the original Bitcoin network.

A.1.2. Scenario B: Reducing the Mining Interval to 100 Seconds (1 MB, 100s)

Bitcoin Architecture: Shifting the mining interval down to 100 seconds exponentially exacerbates the fork rate under traditional mechanics. To maintain stability, the 1 MB block propagation threshold must be strictly compressed to 0.5 seconds a requirement that contradicts the physical 3 second latency limit inherent to decentralized internet infrastructure.

Proposed Architecture: In our architecture, the physical propagation characteristics of a 1 MB block remain unchanged, yet the structural deadline is extended to 100 × 4 = 400 seconds (an 800-fold improvement over the traditional threshold). Consequently, B-Type forks are substantially suppressed. For immediate consensus validation, the proposed framework possesses a decisive structural advantage: due to the complete decoupling of mempool dependency, the Compact Block propagates directly and continuously, free from the round-trip recovery cycles and mempool-induced stalls that frequently impede Bitcoin's Compact Block Relay. While the exact propagation latency remains subject to real-world network conditions, this structural advantage ensures that A-Type fork rates remain bounded within the limits of the original Bitcoin baseline.

A.1.3. Comparative Assessment of Scenarios

Comparing the two paradigms, Scenario B exhibits clear architectural superiority in terms of usability and systemic stability. By reducing the block interval, the transaction confirmation time becomes faster than that of Bitcoin. Furthermore, because individual blocks carry fewer transactions, the absolute probability of hash collisions is minimized, heavily reducing the over-freezing rate. The only trade-off is throughput, yielding approximately 40 TPS compared to Scenario A’s 70 TPS.

A.2. Convergence to the Ultimate Longest-Chain Consensus

Both the traditional Bitcoin paradigm and the proposed framework share an identical fundamental primitive: following prolonged physical network partitions or severe propagation disruptions, the entire decentralized network eventually converges asymptotically toward a single, unique ledger state via the longest-chain rule. The operational divergence is purely structural rather than conceptual. Crucially, regardless of how dynamically the internal pipeline variables—namely the evaluation window (k), the propagation gate (i), and the difficulty penalty coefficient (η)—fluctuate within the system, all operational outputs are ultimately wrapped seamlessly within the standard PoW block header architecture. Consequently, when a long-range chain competition emerges, the resolution is deterministically dictated by the cumulative POW weight, defined herein as the "PoW Gravity." The proposed pipelined mechanism does not violate or bypass the foundational Nakamoto Longest Chain Rule; rather, it merely functions as an optimized tuning of the validation layer's behavior during active fork synchronization and network convergence regimes. In Bitcoin, when a node discovers a longer chain branch, it must download and validate every single missing block sequentially. In contrast, the proposed architecture introduces the explicit $n_k_declaration$ field. This enables nodes to strategically filter, download, and validate only those blocks marked with $n_k_declaration = True$, optimizing resource utilization during state recovery. Admittedly, due to the significantly higher TPS capacity of the proposed architecture, re-synchronizing after an extended network partition induces a higher computational and data download overhead. However, this is an inherent and universal tax paid by any PoW ledger that scales throughput; it is a manifestation of the core Blockchain Trilemma rather than a localized flaw unique to this paper's design.

APPENDIX B. Mempool Freezing Removal and Consensus-Agnostic Bottleneck Reduction via Dual-Coinbase Gradual Invalidation

The main text adopts a UTXO Freezing mechanism (Section 3.3) in which UTXOs generated during the k −block isolation window is strictly prohibited from being consumed in subsequent blocks. It is important to note that this restriction applies exclusively to inter-block UTXO chaining: a UTXO created in block nn cannot be consumed in blocks $n + 1$ through $n + k$. The original protocol does not prohibit intra-block UTXO chaining, where a UTXO created by transaction A within a block is consumed by transaction B in the same block. This distinction is consistent with Bitcoin's standard behavior and is preserved throughout the present analysis.

While this UTXO Freezing mechanism guarantees logical isolation and prevents cascading invalidation, it imposes a rigid economic constraint: UTXOs created in block n remain locked for approximately $k \times T_{block}$ (e.g., 60 minutes for $k = 6$, $T_{block} = 600$s). This constraint is incompatible with high-frequency DeFi protocols, payment channels requiring rapid capital turnover, and any application demanding sub-hour liquidity. More critically, it structurally caps the permissible value of k. Although raising k expands the latency-hiding window and thereby improves fork resistance, the UTXO Freezing constraint prevents the protocol from realizing its own scalability potential.

This appendix proposes an alternative mechanism that preserves the safety guarantees of Theorems 1, 2, 3 and 4 while eliminating the mandatory inter-block freezing period. The key insight is that the protocol need not forbid the consumption of recent UTXOs; it need only make such consumption economically accountable through the $n_k_declaration$ field.

B.1. Dual-Coinbase Structure

We replace the single coinbase transaction of the $(n + k + 1) - th$ block with a dual structure:

- **Coinbase1 (Draft Reward)**: The nominal reward claim of the $(n + k + 1) - th$ block miner. This is recorded on-chain but is not recognized as spendable value until the $(n + k + 1) - th$ block's validity is finalized at height $n + 2k + 1$.
- **Coinbase2 (Finalized Reward)**: The actual reward that the $(n + 1) - th$ block miner receives. Its value is determined deterministically at height $n + k + 1$ based on the $n_k_declaration$ field of the $(n + k + 1) - th$ block.

B.2. Declaration Criterion Modification

The $n_k_declaration$ field is redefined as follows. Let S_{n-k+1}^{n}denote the set of UTXOs generated by blocks $n - k + 1$ through n, and let T_{n+1} denote the set of transactions included in block $n + 1$.

Define the following, where $|\cdot|$ denotes the cardinality (number of elements) of a set:

$$\rho_{n+1} = \frac{|\{tx \in T_{n+1}: tx \text{ direcly or indirectly consumes } S_{n-k+1}^{n}\}|}{|T_{n+1}|}$$

Here, "indirectly consumes" includes chaining: if transaction A directly consumes a UTXO in S_{n-k+1}^{n} and transaction B consumes a UTXO created by A, then B is counted as an indirect consumer. The declaration value is then determined by:

$$n_k_declaration = \begin{cases} True, & \text{if } \rho_{n+1} \leq X\% \text{ and no faulty transactions exist} \\ False, & \text{otherwise} \end{cases}$$

where $X\%$ is a protocol parameter (e.g., $X = 20$). The choice of a transaction count ratio, rather than a fee volume ratio, is deliberate: it guarantees a strict upper bound on computational complexity during potential state updates, minimizes cryptographic overhead on nodes, and avoids consensus split vulnerabilities that could arise from micro-fee calculations. However, the precise determination and systematic optimization of the exact X value across diverse network scales and transaction densities remain beyond the scope of this paper, necessitating further in-depth investigation in future research.

B.3. Coinbase2 Determination

The determination of $Coinbase2_{n+k+1}$ is executed at the canonical height $n + k + 1$, at the moment when the validity of block $n + 1$ is conclusively resolved by the $n_k_declaration$ field embedded in the header of the $(n + k + 1) - th$ Compact Block. It is essential to distinguish between two temporally separated entities:

- $Coinbase1_{n+1}$: the provisional reward claim recorded in block $n + 1$ at the time of its mining. This is the draft entry that was written on-chain but has remained unspendable pending the declaration outcome.
- $Coinbase2_{n+k+1}$: the finalized reward determined at height $n + k + 1$, which is paid to the miner of block $n + 1$.

Address Invariance. The receiving address of $Coinbase2_{n+k+1}$ is identical to that of $Coinbase1_{n+1}$. The two entities represent the same reward claim, differing only in their temporal stage of finalization. No redirection of funds is permitted.

The following analysis presumes that block n has been declared invalid ($n_k_declaration = False$), since this is the only scenario in which $Coinbase2_{n+k+1}$ diverges from the nominal value of $Coinbase1_{n+1}$.

Let $C_{n+1} \subseteq T_{n+1}$ denote the set of transactions in block $n + 1$ that are logically connected to block n. A transaction $tx \in T_{n+1}$ is said to be logically connected to block n if it directly or indirectly consumes any UTXO generated by block n. The relation "logically connected" is defined as the transitive closure of direct and indirect consumption: if transaction A directly consumes a UTXO of block n, and transaction B consumes an output of A, then B is also logically connected to block n, and this relation propagates transitively through any chain of intra-block dependencies originating from block n.

Under the premise that block n is invalidated, every transaction in C_{n+1} loses its validity, and the fees it contributed to block $n + 1$ are consequently revoked. Furthermore, all outputs of transactions in C_{n+1} are also invalidated, and any transaction consuming such outputs—whether within block $n + 1$ or in any subsequent block that has not yet been finalized—is likewise invalidated.

The determination rule is therefore expressed as:

$$Coinbase2_{n+k+1} = \begin{cases} Coinbase1_{n+1}, & if\ C_{n+1} = \emptyset \\ Coinbase1_{n+1} - \sum_{tx \in C_{n+1}} fee(tx), & if\ 0 < |C_{n+1}| \leq X\% \cdot |T_{n+1}| \\ 0, & if\ |C_{n+1}| > X\% \cdot |T_{n+1}|\ \ or\ a\ faulty\ transaction\ is\ present \end{cases}$$

A faulty transaction is defined as one that fails signature verification, attempts a double spend, or violates script integrity, as in Bitcoin.

Four clarifications are necessary:

- **Timing.** All quantities in the above expression are evaluated at height $n + k + 1$. At this moment, the complete data of blocks $n - k + 1$ through $n + 1$ has been globally synchronized, and the declaration outcome of block n is deterministically fixed. The logical connection of a transaction to block n is assessed with respect to the *pre-invalidation* state of block n's UTXO set; that is, the UTXO that block n would have generated had it been valid.
- **Scope of revocation.** The revocation applies to the fees of the transactions in C_{n+1}. Transactions in $T_{n+1} \backslash C_{n+1}$ remain valid and their fees remain credited to the miner of block $n + 1$.
- **Non-applicability when block n is valid.** If block n is valid, and $Coinbase2_{n+k+1}$ equals $Coinbase1_{n+1}$ without any deduction, irrespective of the composition of C_{n+1}.
- **Cascading invalidation.** The invalidation of outputs belonging to C_{n+1} propagates deterministically to any subsequent block that has consumed such outputs, as those outputs are rendered non-existent by the invalidation of their parent

transactions. This propagation logically follows from the deterministic nature of UTXO consumption validation, as the dependency chain is inherently embedded in the state transition, enforcing consistency without auxiliary protocol overhead.

B.4. Localized Invalidity Propagation and Computational Complexity of the Dual-Coinbase Mechanism

The purpose of this subsection is to show that the Dual-Coinbase mechanism replaces global transaction-dependency management with purely local operations, and that its computational complexity is bounded and scales only linearly with the pipeline window k and the block size, independently of the threshold X. Removing mandatory UTXO freezing does not require a global transaction-dependency graph. Every input UTXO carries its origin block, so recent-state dependence can be evaluated directly from the inputs of the block under consideration. Intra-block UTXO chaining is permitted; the treatment below accounts for it through localized transitive closure.

B.4.1. Notation and UTXO Origin and Block-Level Origin-Height Metadata

For a transaction tx, let

$$I(tx) = \{\, u : u \text{ is an input UTXO referenced by } tx \,\},$$

$$O(tx) = \{\, u : u \text{ is an output UTXO created by } tx \,\}.$$

For a block B_m, let T_m denote the set of transactions included in B_m, and define the block-level input and output UTXO sets by

$$I_m = \bigcup_{tx \in T_m} I(tx), \qquad S_m = \bigcup_{tx \in T_m} O(tx).$$

In a valid canonical block, the elements of I_m are distinct, so $| I_m | = \sum_{tx \in T_m} |I(tx)|$.

The transaction-layer representation of an input reference is the Bitcoin-standard outpoint $(txid, vout)$, and the signature of a transaction is computed exclusively over the transaction data, exactly as in Bitcoin. The mechanism developed in this Appendix B, however, requires that the protocol be able to determine, for any input UTXO, the height of the block containing the transaction that directly created the referenced output. This quantity is called the origin height of the UTXO, and is denoted by

$$origin(u) = h \iff u \in S_h.$$

To supply this quantity without altering the transaction layer, the protocol introduces block-level auxiliary metadata. For each block B_m and each input UTXO $u \in I_m$ the block body contains an auxiliary record

$$meta(u) = origin(u),$$

recorded by the miner when the transaction referencing u is included in B_m. The auxiliary metadata is committed to by a dedicated commitment field in the block header, so that $meta(u)$ is cryptographically bound to the canonical block data and cannot be altered without invalidating the block's Proof-of-Work.

Implementation Note. Relative to the UTXO-Freezing baseline, the Dual-Coinbase mechanism changes the block-level data structure and its associated overhead as follows. Because the origin height of each input UTXO must be recorded as block-level auxiliary metadata, a 10 MB Full Block containing approximately 100,000 input UTXOs incurs an additional 400 KB, corresponding to a 4% increase in block size. Rather than introducing a dedicated commitment field into the block header, the mechanism commits to this metadata through the block's coinbase transaction. Specifically, the auxiliary origin-height records are aggregated into a Merkle root, denoted MetaRoot, which is embedded in the coinbase transaction—under the Dual-Coinbase construction, in Coinbase1—and is therefore indirectly committed to by the existing Merkle_Root in the header via the coinbase transaction ID. A block that omits MetaRoot or commits an incorrect value is rejected at the block-validation layer, so the commitment remains consensus-critical without altering the transaction-layer semantics of ordinary transactions. This design preserves the 113-byte header and the associated PoW preimage length. The 400 KB origin-height metadata is recorded only in the Full Block body and therefore does not affect the Compact Block size. The MetaRoot commitment adds only a 32-byte value plus minimal script/output overhead to the coinbase transaction, which is part of the Full Block but not part of the Compact Block; consequently, the Compact Block size remains unchanged. Accordingly, Table 1 and Table 2 need not be revised to include a new header field, whereas Table 3 should be updated to document the auxiliary origin-height metadata and the coinbase-based MetaRoot commitment, so that the UTXO-Freezing baseline and the Dual-Coinbase configuration are clearly distinguished.

This separation of concerns between the transaction layer and the block layer yields three structural properties that are essential to the mechanism developed in the remainder of this Appendix B.

Property 1 (Transaction-Layer Stability under Reorganization). The outpoint $(txid, vout)$ and the transaction signature are invariant under blockchain reorganization. If a transaction tx is re-mined from block H_1 into block H_2, its $txid$, its output indices, and its signature remain unchanged. Consequently, any transaction that consumes an output of tx continues to reference the same outpoint, and its validity is unaffected by the relocation of tx. Only the block-level auxiliary record $meta(u) = origin(u)$ is updated to reflect the new enclosing height. This property preserves the same reorganization stability that Bitcoin's consensus-level outpoint representation provides, and it prevents the cascading invalidation that would arise if the origin height were embedded within the input reference itself.

Property 2 (Compatibility with the Existing Transaction Infrastructure). Because the transaction-layer representation of an input is unchanged, existing wallet software, transaction serialization formats, signature verification procedures, address schemes, and SPV client logic remain fully applicable. The protocol introduces a new block-level validation rule, but it does not alter the semantics of any transaction-level object. This property is a prerequisite for the backward-compatible upgrade path claimed in the main text.

Property 3 (Deterministic Verifiability of the Auxiliary Metadata). A node that maintains the canonical UTXO set can verify $meta(u)$ directly: for each input $u = (txid, vout)$ referenced by a transaction in B_m, the node retrieves the UTXO entry associated with u from the UTXO set and confirms that its recorded origin height equals $meta(u)$. Any discrepancy renders the block invalid. The verification is a bounded lookup per input and introduces no additional asymptotic cost. A node that does not maintain the UTXO set may still verify $meta(u)$ indirectly by reconstructing the relevant UTXO entry from the canonical chain, at the cost of additional storage or bandwidth, in the same manner that Bitcoin nodes verify spent outputs.

For a UTXO u, the origin-height metadata supplies the quantity $origin(u)$ required by the recent-UTXO set

$$R_{m,k} = \{\, u : origin(u) \in [m-k,\ m-1] \,\},$$

which is the central object of the declaration criterion, the local frontier propagation procedure, and the complexity analysis developed in B.4.2–B.4.4. The auxiliary metadata therefore suffices to support every operation that the remainder of this appendix requires, without modifying the transaction layer.

Two clarifications are necessary.

First, the auxiliary metadata is consensus-critical but not transaction-critical. A block that omits $meta(u)$ for any input $u \in I_m$, or that records an incorrect value, is rejected at the block-validation layer. A transaction that is valid with respect to the UTXO set remains valid regardless of the metadata recorded by the miner who included it; the transaction's own signature and outpoint semantics are unaffected. The metadata is therefore a property of the block, not of the transaction, and its correctness is enforced at the block level rather than at the transaction level.

Second, in the fallback scenario discussed in B.4.3, where the Full Block data of an invalidated block B_n is unavailable, the auxiliary metadata recorded in subsequent blocks still identifies inputs whose recorded origin height equals n. This is precisely the condition used in the modified seed-set construction and it permits the local frontier propagation to proceed without requiring the transaction data of B_n. The auxiliary metadata therefore preserves the availability-independent character of the invalidity-propagation procedure.

In summary, by locating the origin-height information in a block-level auxiliary field rather than in the transaction input, the protocol retains the reorganization stability and infrastructure compatibility of Bitcoin's outpoint model while providing the deterministic origin information required by the declaration, frontier-propagation, and complexity analyses of B.4.2–B.4.4.

B.4.2. Transaction-Level Recent-Consumption Ratio

The transaction-level recent-consumption ratio for B_m must account for both direct and intra-block indirect consumption. Define the direct-consumer seed set by

$$A_m^{(0)} = \{\, tx \in T_m : I(tx) \cap R_{m,k} \neq \emptyset \,\}.$$

To capture intra-block chaining, compute the transitive closure within T_m:

$$A_m^{(r+1)} = A_m^{(r)} \cup \{\, tx \in T_m : I(tx) \cap \Big(\bigcup_{tx' \in A_m^{(r)}} O(tx') \Big) \neq \emptyset \,\}.$$

Iterate until the fixed point is reached, i.e., until the minimal r satisfying

$$A_m^{(r+1)} = A_m^{(r)},$$

and set

$$A_m^{(\infty)} = A_m^{(r)}\ .$$

The transaction-level ratio is then

$$\rho_m^T = \left|A_m^{(\infty)}\right| / \left|T_m\right|.$$

The ratio ρ_{n+1} introduced in B.2 is exactly the transaction-level ratio ρ_m^T evaluated at $m = n + 1$:

$$\rho_{n+1} = \rho_{n+1}^T = \left|A_{n+1}^{(\infty)}\right| / \left|T_{n+1}\right|,$$

where the seed set is taken with respect to

$$R_{n+1,k} = \{\, u : origin(u) \in [n - k + 1,\ n] \,\} = S_{n-k+1}^n.$$

Thus the present formulation is not a new criterion but the same declaration rule expressed in origin-based notation, with intra-block chaining handled by the local transitive closure $A_{n+1}^{(\infty)}$.

B.4.3. Local Frontier Propagation of Indirect Effects

Suppose that some transactions of B_n are subsequently declared invalid. Let $A_n (\subset T_n)$ denote the set of transactions in B_n that are invalidated. Define the initial affected-output frontier by

$$F_n = \bigcup_{tx \in A_n} O(tx).$$

For each subsequent block $B_{n+j} (0 < j \le k)$, define the affected transaction set A_{n+j} and the next frontier F_{n+j} recursively. First collect the direct consumers of the previous frontier:

$$A_{n+j}^{(0)} = \{\, tx \in T_{n+j} : I(tx) \cap F_{n+j-1} \neq \emptyset \,\}.$$

Then compute the transitive closure within B_{n+j} :

$$A_{n+j}^{(r+1)} = A_{n+j}^{(r)} \cup \{\, tx \in T_{n+j} : I(tx) \cap \left(\bigcup_{tx' \in A_{n+j}^{(r)}} O(tx') \right) \neq \emptyset \,\}.$$

Iterate until the fixed point is reached, i.e., until the minimal r satisfying

$$A_{n+j}^{(r+1)} = A_{n+j}^{(r)},$$

and set

$$A_{n+j} = A_{n+j}^{(\infty)} = A_{n+j}^{(r)}, \qquad F_{n+j} = F_{n+j-1} \cup \left(\bigcup_{tx \in A_{n+j}} O(tx) \right).$$

This definition captures both direct and intra-block indirect consumption within B_{n+j}, so that affected transactions and the output frontier are not deferred to a subsequent block even when intra-block UTXO chaining is permitted.

If the entire block B_n is invalidated and its Full Block data is unavailable, the initial affected-output frontier cannot be enumerated from $A_n = T_n$. In this case, set $F_n = \emptyset$ as an initial computational frontier and identify direct dependencies by using the origin-height metadata of input UTXOs. For each subsequent block B_{n+j}, where $1 \le j \le k$, modify the initial affected transaction set as follows:

$$A_{n+j}^{(0)} = \{\, tx \in T_{n+j} : I(tx) \cap F_{n+j-1} \neq \emptyset \vee \exists u \in I(tx),\ origin(u) = n \,\}.$$

The first condition identifies transactions that consume outputs already included in the propagated frontier. The second identifies transactions that consume UTXOs whose recorded origin height is nn, thereby detecting direct dependencies on the invalidated block even when its transaction data is unavailable.

Starting from this seed set, compute the intra-block transitive closure using the recurrence already defined above:

$$A_{n+j}^{(r+1)} = A_{n+j}^{(r)} \cup \{\, tx \in T_{n+j} : I(tx) \cap \Big(\bigcup_{tx' \in A_{n+j}^{(r)}} O(tx') \Big) \neq \emptyset \,\}.$$

Repeat this operation until the fixed point is reached, and denote the resulting set by $A_{n+j} = A_{n+j}^{(\infty)}$, using the existing notation. Every transaction added during this closure is affected because it consumes an output produced by a transaction already identified as affected, either through direct dependence on B_n, through the previous frontier, or through an intra-block dependency chain.

After the fixed point is reached, update the frontier using the existing recurrence:

$$F_{n+j} = F_{n+j-1} \cup \Big(\bigcup_{tx \in A_{n+j}} O(tx) \Big).$$

The updated frontier is then used to process B_{n+j+1}. Thus, for each block in sequence, the procedure (i) identifies direct consumers of the accumulated frontier and inputs whose recorded origin height is n, (ii) computes the full intra-block dependency closure, and (iii) adds the outputs of all newly affected transactions to the frontier. This procedure is repeated through B_{n+k}, with the resulting affected transaction sets and frontiers retained for the invalidation analysis.

The frontier remains a logical set for dependency tracking and does not represent valid spendable UTXOs. The origin-height condition is used to identify inputs claiming provenance from B_n; the procedure presupposes that B_n has already been declared invalid and does not, by itself, resolve the separate question of data availability or establish the validity of an input UTXO.

No global transaction-dependency graph is required, and no recursive traversal of the entire blockchain is performed; the necessary information is already localized in the provenance of the input UTXOs and in the transaction set of the current block. The frontier F_{n+j} is logical and is used only for invalidity propagation; it is not a set of valid spendable UTXOs.

For any UTXO u in a valid canonical ledger,

$$|\{\, tx : u \in I(tx) \,\}| \leq 1.$$

A transaction may consume multiple UTXOs, $|\, I(tx) \,| > 1$, so the dependency structure may exhibit fan-in, but a single valid UTXO does not generate unrestricted fan-out. This property bounds the breadth of invalidity propagation at each step of the local frontier recursion.

B.4.4. Deterministic Evaluation at the Declaration Point and Complexity Bound

The declaration associated with B_n is evaluated at the protocol-defined declaration point corresponding to the production of B_{n+k}. Under the propagation assumption of the proposed architecture, the Full Block data required for evaluating the relevant historical state has had the preceding pipeline interval to propagate through the network. The declaration therefore does not depend on an arbitrary local mempool snapshot.

For the recent-origin test,

$$origin(u) \in [n-k,\, n-1],$$

so

$$N_{recent} = |\, u \in I_n : origin(u) \in [n-k,\, n-1] \,|$$

and hence the input-level count

$$|I_n \cap R_{n,k}\,| = N_{recent}$$

is uniquely determined by the canonical input set. The transaction-level ratio ρ_n^T is computed from the fixed point $A_n^{(\infty)}$ defined in B.4.2.

The parameters X and k govern distinct dimensions of the mechanism. The threshold X controls the permissible breadth of recent-UTXO dependence, i.e. cascade width. The parameter k controls the temporal evaluation window, i.e. cascade depth. A bound on X is not a proof that invalidity propagates only finitely many blocks; it limits evaluation breadth, not temporal depth.

The computational cost of the above procedure admits a simple upper bound. Let

$$\tau = \max_m |T_m|\,, \qquad \sigma = \max_m |I_m|$$

denote the maximum number of transactions and the maximum number of input UTXOs in any single block. Within each block B_m, the transitive closure $A_m^{(\infty)}$ visits each transaction at most once and each output UTXO at most once, since a valid UTXO cannot be consumed by more than one transaction. Hence the per-block cost is

$$O(|T_m| + |I_m|) \leq O(\tau + \sigma).$$

Across the evaluation window of k blocks, the total cost is therefore

$$C(k) = O\left(\sum_{j=1}^{k} (|T_{n+j}| + |I_{n+j}|)\right) \leq O(k\,(\tau + \sigma)).$$

The threshold X does not appear in this bound, because X is a declaration criterion rather than a computational parameter: the closure must be computed regardless of the value of X. Consequently, the complexity is linear in k and in the block size, and is independent of X. Increasing k increases the cost only linearly, not exponentially, and increasing X has no effect on the cost at all. This confirms that the Dual-Coinbase mechanism remains computationally tractable even for large pipeline windows and permissive declaration thresholds.

The Nature of the X% Threshold as an Economic Criterion. It is essential to distinguish the X% threshold from a protocol-enforced hard cap on the inclusion of recent-UTXO-consuming transactions. The X% value functions exclusively as a declaration criterion: a block whose recent-consumption ratio ρ exceeds X% is declared False, and its miner forfeits Coinbase2 in full. The protocol does not, however, prohibit the inclusion of such transactions. Consequently, the maximum cascade width X% is sustained not by protocol enforcement but by rational miner equilibrium. A miner who includes a transaction that directly or indirectly consumes UTXOs generated within the recent k −block window must necessarily possess the corresponding Full Block and must judge that the source block will be declared True; otherwise, the transaction is invalid and its fee is revoked from the miner's finalized reward. This structural dependency implies that only miners who have received the Full Block and assessed it as valid will include such transactions. Nevertheless, a miner who accepts the penalty may exceed X%, in which case the cascade width of that block exceeds the nominal threshold. The protocol thereby replaces a rigid inclusion prohibition with an economic accountability mechanism: rational miners voluntarily constrain their recent-UTXO consumption to remain within X%, while adversarial or risk-seeking miners may exceed it at the cost of full reward forfeiture. Across the pipeline window of k blocks, the maximum cascade size is bounded by $k \cdot X \cdot B_{max}$ where B_{max} denotes the maximum number of transactions per block. This bound is linear in both k and X, and does not admit exponential amplification: each block independently enforces its own X% constraint, so the cascade does not compound across successive blocks. Consequently, even for any X ($0 < X < 100$), the cascade remains finite and bounded; under rational miner equilibrium, the per-block cascade width does not exceed X. Increasing X therefore does not lead to system collapse, but merely raises the maximum allowable cascade size, which remains a tunable design trade-off between liquidity and cascade risk. The X% threshold thus serves simultaneously as a safeguard against unbounded cascading invalidation and as a parameter governing that trade-off.

Design Implication. This observation yields a direct calibration rule for the X% threshold. Since the maximum cascade size over the pipeline window is bounded by $k \cdot X \cdot B_{max}$, and since the system remains operational for any $X \in (0{,}100)$, the choice of X reduces to a single design question: what magnitude of cascading invalidation is acceptable for the intended application? Given a cascade budget C^*, the threshold is determined directly by

$$X \leq C^*/(k \cdot B_{max}).$$

The calibration of X therefore does not require modeling the stochastic dynamics of the cascade process; it follows deterministically from the pipeline depth k, the block capacity B_{max}, and the tolerable cascade budget C^*. In this sense, X is a design parameter rather than a security constraint.

Consequently, the Dual-Coinbase mechanism replaces mandatory inter-block UTXO freezing with an economically accountable and computationally localized test,

$$origin(u) \in [n - k,\, n - 1],$$

together with bounded local frontier propagation for indirect effects, including intra-block chaining. No global transaction-dependency graph and no recursive traversal of the entire blockchain are required; the necessary information is already localized in the provenance of the input UTXOs.

B.5. Compatibility of the Dual-Coinbase Mechanism with the Existing Analysis

Claim. The Dual-Coinbase mechanism is compatible with the analytical framework underlying Theorems 1–4, subject to the assumptions described above.

Argument. Theorems 1–4 analyze block-level PoW fork competition and miner incentives under the delayed-validation framework. They do not depend on the particular mechanism used to control recent-UTXO dependencies at the transaction-validity layer. In the baseline construction, this control is provided by UTXO Freezing; in the Dual-Coinbase construction, the same risk is controlled through deterministic dependency measurement and economic accountability.

More specifically, UTXO Freezing and Dual-Coinbase address the same underlying problem: a miner may otherwise incur an unintended loss when a block containing transactions that depend on recently generated UTXOs is subsequently affected by delayed validation. UTXO Freezing prevents such exposure ex ante by prohibiting the consumption of designated recent UTXOs. Dual-Coinbase instead permits such dependencies but limits their economic consequences through the ρ-based declaration rule and the conditional realization of the deferred coinbase reward. Thus, the two mechanisms differ in how transaction-level risk is controlled, but neither changes the block-level PoW competition model analyzed in Theorems 1–4.

Theorems 1–4 depend on the following properties of the delayed-validation framework:

- A deterministic and publicly verifiable declaration.

The value of $n_k_declaration$ must be determined by protocol-defined, publicly available information rather than by subjective judgment. Under the Dual-Coinbase mechanism, the dependency ratio ρ and the presence of faulty transactions are evaluated deterministically at the designated declaration point. At height $n + k + 1$, the Full Blocks required for this evaluation are assumed to be fully synchronized under the propagation assumptions of the main construction. Consequently, all nodes can independently derive the same declaration from the same canonical data.

- A fixed declaration point and a well-defined economic consequence.

The Dual-Coinbase mechanism does not change the timing at which the declaration is evaluated. The declaration is still made at the prescribed height, and a False declaration carries the same economic consequence assumed in Theorems 1–4: the corresponding deferred reward is forfeited, while the applicable difficulty penalty η remains in effect. Coinbase1 therefore represents the provisional reward associated with block production, whereas Coinbase2 represents the reward that becomes economically realizable after the relevant validity condition has been resolved.

- Invariance of transaction-layer objects under reorganization.

The transaction-layer representation of an input reference remains the Bitcoin-standard outpoint $(txid, vout)$, and the transaction signature is computed exclusively over the transaction data. Consequently, both the outpoint and the signature of a transaction are invariant under blockchain reorganization: if a transaction is re-mined from one block into another, its $txid$, its output indices, and its signature remain unchanged. The origin-height information required by the Dual-Coinbase mechanism is carried as block-level auxiliary metadata, as defined in B.4.1, and is therefore updated only at the block layer when the enclosing height changes. This separation ensures that the cascading invalidation that would arise if the origin height were embedded within the input reference does not occur, and that the transaction-level semantics assumed by Theorems 1–4 are preserved without modification.

- Preservation of the PoW competition structure.

The Dual-Coinbase mechanism does not alter the block-header PoW, cumulative-PoW chain selection, or the block-level consequences of True and False declarations. Therefore, the strategic situations considered in Theorems 1–4 remain unchanged. In particular, the analysis of ordinary PoW forks in Theorem 1, the incentive against False declarations in Theorem 2, the private-fork success probability in Theorem 3, and the withholding attack considered in Theorem 4 operate on the same declaration and PoW competition structure.

The distinction between the transaction-validity layer and the block-level competition layer is therefore essential. The UTXO Freezing and Dual-Coinbase mechanisms determine how recent-UTXO dependency is controlled and how transaction-level risk is accounted for. Theorems 1–4, in contrast, determine how miners compete once the resulting validity and declaration outcomes are defined. Replacing UTXO Freezing with Dual-Coinbase changes the former layer but does not change the latter, provided that the replacement preserves deterministic declaration, the prescribed declaration timing, public verifiability, and the economic consequences assumed by the theorems.

Accordingly, the proofs of Theorems 1–4 remain applicable to the Dual-Coinbase construction without modification at the level of their PoW fork-competition arguments. The Dual-Coinbase mechanism should therefore be understood not as a

modification of the underlying PoW security model, but as an alternative transaction-validity and economic-accountability mechanism that implements the same delayed-validation framework.

B.6. Liberation of the Pipeline Parameter k and Economic Analysis of Voluntary Freezing

The removal of mandatory UTXO freezing has a profound structural consequence: it liberates the pipeline parameter k from its previous economic constraints.

In the main text, k must satisfy two conflicting requirements:

- Large enough to hide the Full Block propagation delay ($k \cdot T_{block} > T(B_n)$).
- Small enough to keep the UTXO freezing duration economically tolerable.

The latter constraint caps k at approximately $k = 6$ under a 600-second block interval. This cap, in turn, limits the protocol's resilience to propagation anomalies, nested forks, and Data Withholding Attacks.

With UTXO freezing removed, the second constraint vanishes. The parameter k can now be increased freely to $k = 10$, $k = 30$ or beyond, subject only to the requirement that the declaration window remains computationally verifiable. The consequences are substantial:

- **Fork resistance:** The probability of a nested fork P_3 decreases geometrically with k (Table 9), rendering the consensus effectively immune to propagation-triggered divergence.
- **Data Withholding resilience:** A larger k provides honest nodes with a proportionally longer window to obtain the withheld Full Block before the declaration deadline, substantially neutralizing withholding-based attacks.
- **Design space expansion:** Block size, block interval, and k can now be jointly optimized without the UTXO freezing constraint. The protocol thereby provides a viable framework to explore the three-dimensional design space. This space—balancing block size, block interval, and pipeline depth—was structurally constrained under mandatory freezing models.

Economic Analysis of Voluntary Freezing. Although the protocol no longer forbids inter-block consumption of recent UTXOs, rational miners will voluntarily avoid such consumption in most cases. The reasoning is straightforward. A miner who includes a transaction that directly or indirectly consumes UTXOs generated within the recent $k -$ block window faces a nonzero probability that the source block will eventually be declared invalid. If that happens, the transaction loses its validity, and the fee it contributed is revoked from the miner's finalized reward. The miner must therefore weigh the immediate fee revenue against the expected loss arising from this revocation risk.

Consequently, voluntary inter-block freezing emerges as the equilibrium behavior without any need for protocol enforcement. The freezing that was previously mandated by rule is now sustained by economic gravity.

High-frequency DeFi transactions, however, may offer sufficiently high fees to compensate for the residual risk. In such cases, the miner faces a straightforward trading problem: comparing fee revenue against expected penalty. If the fee is large enough to outweigh the expected loss, the miner includes the transaction; otherwise, the miner excludes it. This transforms a protocol-level prohibition into a market-based allocation mechanism, allowing urgent transactions to clear at a premium while ordinary transactions remain safely frozen by economic gravity. The result is a flexible equilibrium in which liquidity is allocated according to willingness to pay, without compromising the safety guarantees established in Theorems 1–4.

B.7. Applicability to Non-PoW Consensus and Consensus-Agnostic Bottleneck Reduction

The mechanism proposed in this appendix is not specific to Proof-of-Work. It arises from the structural decomposition of block production into two components: (i) the propagation of a minimal Compact Block, and (ii) the background availability of Full Block data. This decomposition is independent of the mechanism used to elect block producers.

In Proof-of-Stake, block production is determined by stake weight rather than hash power. The asynchronous pipeline, the dual-coinbase structure, and the Compact Block propagation apply identically. The $n_k_declaration$ field remains a deterministic function of publicly available data, and the only change is the source of block production authority.

In Byzantine Fault Tolerance protocols, block production is determined by validator voting. The asynchronous pipeline can be layered atop the BFT finality gadget: validators produce Compact Blocks at each height while Full Block data propagates in the background. The $n_k_declaration$ field can be integrated into the voting round without altering the determinism or transparency of the declaration.

In all cases, the remaining bottleneck is the same: the physical propagation latency of the Compact Block. The consensus algorithm determines who produces blocks and how they are ordered; the proposed mechanism determines how fast the resulting chain can propagate. These are orthogonal concerns, and the mechanism is therefore consensus-agnostic.

It follows that the true bottleneck of a blockchain is not the consensus algorithm but the physical propagation latency of the Compact Block. All other factors—Full Block propagation, mempool synchronization, UTXO finality constraints, fork convergence latency, and declaration overhead—are either absorbed into the PoW computation time or eliminated by the dual-coinbase structure. The Compact Block, containing only the block header, the input UTXO count, and the compressed UTXO identifiers, represents the information-theoretic minimum required for a node to verify PoW, validate the declaration field, and initiate mining of the next block. No further compression is possible without sacrificing verifiability, and the propagation latency of this minimal payload is bounded below by the physical limits of the network medium. This constitutes the theoretical lower bound of blockchain scalability, and it holds regardless of whether the underlying consensus is PoW, PoS, BFT, or DAG-based. The mechanism thus establishes a general architectural principle: the scalability of a blockchain is ultimately determined by the physical speed at which its minimal consensus payload can traverse the network.