SoK: Multiple Concurrent Proposer Blockchains

Benjamin Marsh[1,2], Alejandro Ranchal-Pedrosa[1], and Max Resnick[3]

[1] Sei Labs
[2] University of Portsmouth
[3] Anza

Abstract. Multiple concurrent proposer (MCP) protocols let several parties contribute transactions to the same ledger update. They can distribute communication and reduce one proposer's control over inclusion and order, but these benefits depend on how the contributions are combined. We describe a common architecture from transaction receipt through proposal formation, agreement, data recovery, ordering, execution, and state finality. Within it, we identify four direct MCP families: fixed proposal vectors, selected proposal collections, persistent proposal streams, and causal proposal DAGs. We compare them with constrained assembly, where several parties restrict a block assembled by one party. Across 39 configurations, we distinguish agreement and progress from service for a particular transaction, and compare hiding, ordering, communication, throughput, and latency at explicitly named endpoints. The comparison shows why additional proposers alone do not guarantee inclusion, why hiding does not settle who can react last, and why parallel disseminations need not improve useful execution. It also identifies the recovery, payment, and organizational assumptions needed to turn independent proposal rights into reliable transaction service.

1 Introduction

For many years, research on practical Byzantine fault-tolerant consensus has sought secure protocols with ever lower latency and higher throughput [22,115,31,23]. However, high throughput and low latency, while important for financial applications, are not the only properties one might want in a financial exchange [89]. Protocol designers are therefore expanding the requirements for blockchains built for financial applications to include censorship resistance, determinism [21], fairness [52], predictability, and hiding [41].

To see why these properties matter, recall that Bitcoin was originally designed as a form of electronic cash [75]. Its successors, such as Ethereum, enabled more complex financial applications, including on-chain automated market makers [29] and lending markets [89]. In these applications, transaction timing and ordering can affect financial outcomes, creating opportunities for speed races and miner extractable value (MEV), together with the problems associated with them [29]. Once transaction timing and ordering have economic value, those who control them can extract rents. For the purposes of this article, we focus on leaders

in leader-based proof-of-stake consensus, where successive leaders exercise a temporary, serial monopoly over transaction inclusion and ordering [41,78].

Multiple concurrent proposer protocols are a promising direction for achieving these goals [41]. Blockchain research groups are developing both theoretical constructions and practical protocol designs intended for deployment on their chains [21,5]. However, these efforts adopt different definitions and prioritize different desiderata [41,21]. The resulting protocols can therefore differ substantially even when presented as solutions to similar problems, making their claims difficult to compare and debates about their relative merits difficult to resolve.

The purpose of this systematization is to bring these streams of work into a common framework and specify precisely which services and guarantees each provides. By making their guarantees explicit, we seek to reduce confusion about the similarities and differences among these protocols.

To motivate the properties at issue, we adapt the auction example in *Multiple Concurrent Proposers: Why and How* [41, Section 1.1]. Consider an on-chain auction for a lot of an asset with a common value to bidders. This value evolves over time with a common external signal, such as the asset's observable price on an external trading venue. The auction therefore takes place on chain while the value of the item being sold changes with conditions outside it. This stylized setting captures features of several on-chain financial applications, including lending-market liquidations [89], trading venues [29], and perpetual futures markets, where transaction outcomes depend on changing external prices.

We use this setting to compare three protocol configurations and an idealized fourth limit. The first is a single-proposer baseline: one proposer controls the slot and, subject to the protocol's validity constraints, chooses which transactions to include and how to order them. This models the proposer discretion present in leader-based block production in systems such as Ethereum and Solana [89,78].

A bidder can win the item only if its bid is included in the auction. The leader can therefore exclude every competing bid and include only its own. The leader obtains the item for zero and captures its entire common value as rent [41, Section 1.1].

The first configuration fails because the leader can exclude competing bids. Suppose we could somehow remove the leader's power to exclude competing bids.[4] Would that fix the issue? Unfortunately, no. With two bidders and public bids, the later bidder can observe the earlier bid and an updated external signal. Whenever the common value exceeds the earlier bid, the later bidder can outbid it by an arbitrarily small amount and win the item. When the common value falls below the earlier bid, the later bidder can refrain from submitting a competing bid. The earlier bidder therefore loses the profitable opportunities and wins only when the value moves against it, exposing it to extreme adverse selection.

If the bidders are risk neutral and the common value evolves according to a martingale, we can solve the model by backward induction. The earlier bidder chooses not to participate, so the auction again clears for zero. The last mover,

[4] For now, we abstract from how this inclusion guarantee is enforced. We return below to multiple concurrent proposer mechanisms for providing it.

typically the leader or a party controlling it, captures the common value of the item as rent [41, Section 1.1, Scenario 2].

Censorship resistance alone therefore does not resolve the problem in this setting. In the third configuration, we retain censorship resistance and add pre-trade privacy[5]: even if the second bidder acts slightly later, it cannot observe the first bidder's bid before submitting its own [41].

Pre-trade privacy nevertheless leaves an informational advantage: the later bidder may still observe a more recent external price. A leader with this last-look advantage can therefore continue to earn an information rent. Moallemi, Pai, and Robinson characterize this rent [73, Section 3.2]. In their baseline geometric Brownian motion model, the later bidder's expected rent tends to zero as its timing advantage tends to zero, and expected auction revenue approaches the common value observed when the earlier bidder bids [73, Sections 4 and 5.1].

In an idealized fourth configuration, we retain censorship resistance and pre-trade privacy while taking this last-look interval to zero. This removes the information rent due to the later observation in the model, without requiring the external asset price to stop evolving. The protocol-design objective is therefore to combine safety and liveness, high throughput and low latency, censorship resistance, and hiding with as small a last-look advantage as possible.

This idealized benchmark gives us a clear design objective, but comparing proposed protocols requires greater precision. Censorship resistance and pre-trade privacy have different definitions and degrees of protection across these designs [41,8]. The last-look advantage can also differ substantially from one protocol to another. Protocol analyses often focus on establishing censorship resistance and hiding, while giving less attention to minimizing the last-look advantage. Understanding what each protocol actually provides therefore requires comparing these guarantees and the timing of participants' information and actions on common terms.

In a conventional blockchain, the single proposer of a slot assembles the candidate block and submits it to validators for agreement. This position lets it extract MEV or delegate construction to a builder [29]. Private order flow, relays, builder concentration, and external censorship pressure shape who controls the candidate [111,47,46]. Multiple concurrent proposer (MCP) protocols give several parties proposal rights for the same ledger update. These parties can disseminate transactions in parallel and offer users alternative routes into the ledger.

Combining these proposals into one ledger update requires further decisions. Validators still need to agree on which contributions count, obtain the selected data, and derive a common execution order. A transaction sent to several proposers may appear several times. Two different transactions may each be valid when submitted but compete for the same balance or liquidity when executed. A silent proposer and a proposer that withholds the body of an already selected

[5] Pre-trade privacy can be provided by cryptographic hiding mechanisms, such as threshold encryption or appropriately designed commit-reveal schemes. We discuss these mechanisms below.

proposal create different recovery problems. Concurrency also gives participants opportunities to copy or react to one another's proposals [78,67].

This systematization follows concurrent proposals from submission to a canonical ledger update and compares the guarantees and costs along that path. Here, *canonical* means accepted as part of the common ledger under the protocol's agreement and finality rules. Our starting point is an architecture that follows a transaction from receipt to its effect on state. This makes it possible to locate a guarantee precisely: a proposal can be selected before its body has been recovered, and a transaction can be ordered without its execution succeeding.

We classify protocols by the rule that determines which proposed contributions become canonical. Four families directly combine independent proposals: fixed proposal vectors, selected proposal collections, persistent proposal streams, and causal proposal DAGs. The last use a directed acyclic graph (DAG), in which proposals refer to earlier proposals and those references help determine membership or order. We also compare constrained assembly, where contributors impose requirements on a block assembled by one leader or builder. The distinction depends on what the protocol selects, rather than on whether its messages happen to travel through a graph.

Prior work supplies several of the results needed for this comparison. Garimidi, Neu, and Resnick study selective censorship resistance and hiding in one MCP construction [41]. Alpos et al. compare broad protocol families using censorship and performance metrics [2]. Elsheimy et al. and Saraf et al. establish tradeoffs within transaction-assignment and rational-bribery models [36,91]. We connect these results through a common description of proposal selection and the later work needed to execute the selected transactions. The source's assumptions and endpoint remain part of each claim.

The comparison yields five findings. (i) **More proposers do not by themselves guarantee inclusion.** Inclusion needs an accessible, protected path; an enforced inclusion rule can also constrain a single assembler. (ii) **Hiding and inclusion address different powers.** Concealment limits reactions; inclusion limits omission. Combining them requires recovery and reveal rules [41]. (iii) **Deterministic ordering leaves earlier choices open.** A fixed merge leaves earlier influence through selection, timing, fees, and dependencies. (iv) **Parallel proposals can move the bottleneck.** Faster dissemination can leave recovery, duplicate work, or shared-state execution as the bottleneck. (v) **Distinct proposal paths can share the same controller.** Several proposers may depend on one builder, relay, or decryption operator.

The framework connects proposal reconciliation to transaction service. Section 2 introduces the architecture and properties; Section 3 develops the protocol families; and Section 4 compares the configurations. Section 5 draws out their guarantees and tradeoffs. Sections 6 and 7 follow the remaining work through recovery, execution, payment, and control over proposal paths. The appendices give the evidence, review method, and relation to prior work.

2 Architecture and Comparison Framework

2.1 From a client transaction to a shared ledger

A blockchain maintains a ledger and the application state produced by that ledger. Clients submit transactions, such as payments or contract calls. *Proposers* collect transactions and propose contributions to the ledger. *Validators* check protocol messages and participate in agreement on the ledger's next update. Validators commonly also execute transactions to maintain a local copy of the state. The replication literature often calls these participants *replicas*. These are roles: the same participant can propose, vote, store data, and execute. A *leader* coordinates a particular agreement instance; it need not be the only proposer. A *builder* constructs a proposal or payload on behalf of a proposer.

Figure 1 separates five functions common to the protocols we compare. **Admission and proposal formation.** A proposer receives transactions, applies its admission policy, and places accepted ones in a candidate. **Data dissemination and recovery.** Participants distribute proposal bodies and retrieve or reconstruct missing data. Bodies can travel in full or as coded fragments, enough of which reconstruct the original. **Agreement and reconciliation.** Validators agree which contributions form the next ledger update; we call the rules for combining them *reconciliation*. **Transaction ordering and filtering.** The protocol derives an execution sequence from the selected contributions, determining transaction order and handling duplicates and replays. **Execution and state finality.** Validators apply that sequence to the prior state using a common execution context, including the applicable block metadata and fee parameters. Some systems then separately attest the resulting state or its digest. These functions need not be separate software components or consecutive consensus rounds. Agreement and data dissemination can overlap, and a proposal reference can be finalized before every validator has its body. Where a system delegates execution to separate nodes, we identify that role explicitly. Otherwise we use *validator* throughout.

We call one protocol-defined reconciliation event a *ledger step*. It fixes the new canonical contributions for one ledger update or log segment. A conventional block at a height is one example. Other protocols agree on several proposals together or on how far to advance several continuing proposal histories. A step can therefore span several network rounds or component decisions. The terms *slot*, *height*, and *cut* name particular protocols' units; *ledger step* lets us compare their outputs without assuming identical timing.

2.2 Proposals, eligibility, and missing data

A *proposal* is an authenticated record that identifies its author and the contribution it offers. Its *body* contains transactions and the additional data needed to validate or execute them. A *reference* binds the record to that body, for example through a cryptographic digest. The record may also identify a slot or sequence number and carry ordering metadata. Receiving a reference tells a validator which body is meant; receiving or reconstructing the body gives it the data.

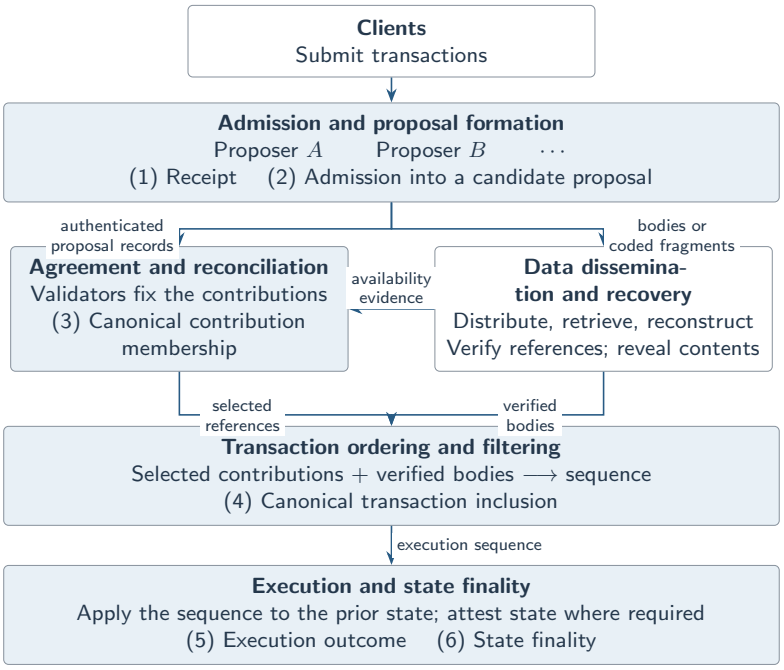

Fig. 1. A common architecture and six transaction checkpoints. MCP adds independent proposal paths at the top; the reconciliation rule determines how their contributions become canonical. Arrows show dependencies, not mandatory message rounds. Data recovery and agreement may overlap; reveal is needed only for hidden contents.

Transaction eligibility and proposal eligibility apply to different objects. An *admissible transaction* satisfies the premises of the particular service claim, such as receipt before a deadline, an adequate fee, available capacity, and specified state conditions. This is a condition on the transaction and its submission. It is not a proposal status. For proposals, we distinguish the following checks: (i) **Properly formed.** The record satisfies authentication and format rules. (ii) **Statically valid.** It passes the application checks that can be made against its declared context without executing the other selected transactions. Hidden contents may postpone some checks until reveal. (iii) **Qualified.** It satisfies the protocol's selection requirements, including any required receipt attestations or certificates and the checks applicable at that stage. These checks do not establish that every enclosed transaction will succeed. Two transactions can each pass a check against the same initial balance, while only one can spend it after ordering.

Missing and *unavailable* describe different failures, rather than additional levels of validity. A *missing proposal* means that an expected proposal path has supplied no acceptable record by its cutoff. The protocol might wait, leave an empty entry, omit that candidate, switch to a fallback procedure, or retain an earlier contribution from that path. An *unavailable body* means that a reference

has been selected but a validator does not yet have the matching data. The protocol must then specify how to retrieve or reconstruct those data and handle stalled recovery. A validator cannot simply skip the body after its own timeout: that could give different validators different execution sequences.

An *availability certificate* is a set or aggregate of authenticated attestations to a protocol-defined availability condition. For example, if each signer attests possession of a full body and at most f signers are faulty, $f+1$ distinct signatures imply at least one correct holder under a static fault model. A certificate for coded fragments instead needs enough recoverable fragments to reconstruct the body. In either case, recovery depends on what signers checked, how long data are retained, who serves them, and the network assumptions. The certificate is evidence for those obligations; it is not the body itself.

2.3 Multiple concurrent proposal

A ledger step is native to the protocol when its reconciliation rule determines which proposal contributions become newly canonical. The rule may select candidates directly, advance continuing proposal histories, or select a graph region through an anchor. Grouping separate ledger decisions afterwards does not create a native step. For an execution ξ, let $\mathcal{P}_s(\xi)$ be the proposal records selected at native step s, including records selected through binding references. Write $\mathsf{author}(p)$ for the authorized proposal path of record p, including its lane or instance where applicable. Transaction-carrying proposals include bodies represented by ciphertexts, coded fragments, or binding references.

Definition 1 (Multiple concurrent proposal). *A protocol configuration provides* multiple concurrent proposal *(MCP) if it admits an execution ξ under its stated model, a protocol-valid native decision at ledger step s, and transaction-carrying proposal records p, q such that*

$$\{p, q\} \subseteq \mathcal{P}_s(\xi), \qquad \mathsf{author}(p) \neq \mathsf{author}(q),$$

and the following conditions hold.

(i) ***Independent proposal authority.*** *Each author is authorized to form its complete candidate record without requiring the other candidate to exist or acquiring the other's exclusive proposal authority.*

(ii) ***Concurrent eligibility.*** *In ξ, the two rights to form candidates for step s overlap before its decision, and the resulting contributions can both qualify under its selection rule.*

(iii) ***Joint canonical contribution.*** *Both records remain separately identifiable constituents of the selected output. Their bodies are required inputs to the protocol's subsequent recovery, ordering, and filtering rules, rather than merely evidence constraining a separately assembled payload.*

The definition concerns permitted joint contribution, not guaranteed inclusion of every proposal. Concurrency concerns proposal rights, not simultaneous transmission or receipt. Distinct proposal identities can share an economic controller.

Duplicate transactions may be filtered, and conflicts may fail during execution. In a DAG, independently formed proposals can become canonical together through a later anchor. Membership retains the configuration's stated finality assumptions and error probability.

Several transaction submitters, builders competing for one selected payload, or inclusion lists constraining one assembler do not alone meet the definition. Nor do competing equivocations for one position, successive exclusive leaders grouped after confirmation, or shards updating disjoint ledgers merely because they operate in parallel. The boundary comparisons retain their separate scope labels.

2.4 Six checkpoints for a transaction

We use six checkpoints to say exactly how far a guarantee or measurement reaches. The numbering matches Figure 1. **(1) Receipt.** An eligible proposer or ingress participant has received the client's transaction. A claim must state who receives it: receipt by one correct proposer and receipt by every correct validator are different premises. **(2) Proposal admission.** A proposer places the transaction in an authenticated candidate. Admission names this proposer choice; *selection* names the protocol's later choice among contributions. **(3) Canonical contribution membership.** The body or its reference becomes a required contribution to a ledger step. Recovery, reveal, and transaction positions may remain unresolved. **(4) Canonical transaction inclusion.** The transaction enters the canonical execution sequence after recovery, ordering, and filtering. One retained copy suffices if several selected proposals carry it. **(5) Execution.** A validator processes the transaction against the state left by preceding transactions, producing success, rejection, or contract reversion. **(6) State finality.** The resulting state transition or commitment obtains the protocol's required finality, including separate state attestation where applicable.

Receipt does not imply proposal admission: a proposer may queue or omit a transaction. Admission does not imply selection: a valid candidate may lose to other candidates or miss a deadline. Selecting a reference does not by itself ensure that its transactions become executable: the body may still need recovery or reveal. Finally, inclusion gives a transaction a place in the sequence, not a promise that the application's requested action succeeds.

Filtering at the fourth checkpoint covers *deduplication*, which retains the protocol's prescribed copy when the selected input repeats a transaction, as when a user submits through several proposers for reliability, and *replay filtering*, which applies the protocol's rules to transactions already processed in earlier ledger history. Both need a precise transaction identity (Appendix B); different signed transactions spending the same balance are conflicts, not necessarily duplicates.

For example, suppose proposers A and B both receive and propose a payment x. Agreement selects both contributions. Once their bodies are available, the ordering and duplicate rules retain one copy of x. Its execution can still fail if an earlier transaction has spent the required funds. The payment has reached inclusion in that case, but has not succeeded.

The checkpoints describe service, not six mandatory serial rounds. *Ordering finality* is the point at which the canonical transaction order becomes irreversible under the stated model. Local execution may lag that order. A protocol may derive state finality from final order and deterministic execution rather than run another certification round. For probabilistic finality, the claimed confidence or error probability is part of the endpoint. We also track body retrievability and plaintext visibility separately, since either can precede or follow selection.

2.5 Scope of the comparison

Our comparison focuses on protocols deployed in, or under consideration for implementation in, real blockchains, together with the earlier protocols from which these designs descend. We prioritize this implementation-oriented literature and its lineages rather than aiming for an exhaustive survey of theoretical constructions. Consequently, proposals with no identified path to implementation may receive less attention, except where they clarify the origins or guarantees of the designs we study. We compare the guarantees provided by each configuration regardless of whether it is labeled MCP.

Hiding and fair-ordering mechanisms are *overlays*: either can be added to a system with one or several proposers. We credit them only when the compared configuration actually includes them.

The comparison covers 39 configurations, using papers, specifications, and measurement reports available in a source snapshot frozen on 9 September 2026. A configuration is a specific protocol version and composition, rather than a project name. Appendix A explains how the sources were selected, how claims were recorded, and the review's limitations.

2.6 Properties used in the comparison

Each paragraph defines one property and its comparison criteria. Guarantees retain the source's adversary and network model. In committee protocols, n counts validators and f bounds Byzantine validators, which may deviate arbitrarily; a *correct* validator follows the protocol. Byzantine fault tolerance (BFT) means preserving the stated guarantees despite these faults. We distinguish fixed (static) corruption from corruption during execution, and report stake and hash-power thresholds separately.

Safety. Correct validators do not finalize incompatible ledger outputs; their finalized histories remain consistent as they grow. We record the protected output, fault threshold, adversary, authentication, and confirmation error. Agreement on references extends to state only with the required common prior state, input order, execution context, and execution semantics.

Liveness. Correct validators continue to produce ledger decisions under the stated operating conditions. We record the fault bound, network model, data and randomness requirements, and termination or delay claim. *Sync* requires a known delivery bound; *PSync* (partial synchrony), timely delivery after an unknown global stabilization time (GST); and *Async*, randomized eventual progress without

a fixed delivery bound. Ledger progress alone does not guarantee service for a particular transaction. Where the claimed endpoint requires selected bodies, progress also requires their recovery under the stated certificate, custody, retention, reconstruction, and corruption assumptions. We record whether one missing body stalls its contribution or the whole step; eventual retrieval alone gives no deadline.

Inclusion and censorship resistance. A specified transaction or proposal is protected from omission after meeting stated receipt and admissibility premises. *Bounded* inclusion reaches a named checkpoint within a stated horizon; *eventual* inclusion has no fixed horizon. *Include-or-empty* protects the input whenever the output is nonempty but permits empty steps. We record the protected object, receipt threshold, timing, capacity, endpoint, horizon, and failure probability. Economic deterrence is separate. Appendix B.4 formalizes these distinctions.

Content hiding. A specified observer or coalition cannot distinguish admissible contents before a named event, beyond permitted leakage. We record the observers, reveal event, security notion, and exposed metadata, including whether the receiving proposer sees plaintext. A bound on bits leaked by coded fragments does not by itself establish semantic hiding until reconstruction [86].

Protection against last look. A named actor cannot use a later observation to revise a protected choice after another participant's corresponding choice closes. We record the actor, information, choice, and *binding point*, after which that choice cannot change. Creating a body, selecting among committed candidates, and releasing a decryption share are different choices. Hiding also leaves reactions to later public prices possible unless the relevant action has closed [41,8].

Ordering and fairness. An ordering rule derives an execution sequence from selected inputs; a deterministic rule gives the same sequence for the same inputs. Fairness additionally constrains relative order by a criterion such as observed receipt order. We record who controls inputs and metadata, whose observations count, the support threshold, and tie, cycle, and batch rules. Fixed order after selection leaves earlier selection and timing choices open [52,19,51].

Communication cost. The data transmitted to complete a named task or reach a named endpoint. We record bytes or messages, aggregate or per-participant cost, payload and committee sizes, batching, and faults. Dissemination, agreement, reveal, and recovery costs retain any omissions in the source. An object-size cap is not a bound on network traffic.

Throughput and executed goodput. Throughput counts work completed per unit time at a specified checkpoint. Executed goodput counts distinct useful transactions or state transitions, excluding redundant copies and failed conflicts under the workload's definition of useful work. We record endpoint, unit, workload, hardware, network, and faults. Ordered bytes, executed gas, and successful transactions are different measures.

Latency. The elapsed time from a named start to a named endpoint. User latency is $L_E(x) = t_E(x) - t_{\text{submit}}(x)$. We record both events, analytical or measured status, load, faults, and distribution or percentile. A bound from an available proposal excludes earlier queueing and dissemination; ordering, execution readiness, execution, and state attestation are different endpoints.

Cadence. The interval between successive starts of a named production or decision unit, such as proposals, slots, cuts, or DAG rounds. We record the unit, start rule, and reported interval. Pipelining can start new units before earlier ones finish, so cadence does not determine latency.

Appendix A.1 defines the coding conventions for these properties, and Appendix B formalizes the architecture.

3 Proposal Reconciliation

The architecture in Section 2 separates the stages a transaction passes through; we now use the reconciliation stage to organize the protocol design space. Reconciliation fixes the canonical contributions at checkpoint 3 of Figure 1. We distinguish four direct MCP families by the rule that selects them.

Two families select individual candidates for a ledger step. A **fixed proposal vector** reserves labeled output positions for the designated proposers, as in Cadence [8]. A **selected proposal collection** chooses among eligible candidates without reserving those positions, as in HoneyBadgerBFT [71].

The other two families select contributions from continuing histories. **Persistent proposal streams** advance and merge prefixes of separate proposal histories, as in Autobahn [42]. **Causal proposal DAGs** use references across authors in a shared graph to determine canonical membership or order, as in Narwhal-based protocols and Mysticeti [31,100,7]. The distinction follows the protocol rule: a decision over stream prefixes belongs to the stream family even when encoded as a vector. References used only for transport or to rank stream outputs leave that classification unchanged. Separating dissemination from agreement is compatible with either family. Narwhal supplies the certified graph used by causal DAG protocols and belongs with that family.

We compare these four families with **constrained assembly**. Here, other participants impose requirements on a block assembled by one leader or builder, as in FOCIL [105]. Their constraints affect the block's eligibility without becoming independently selected proposal bodies. This is MCP adjacent. Figure 2 illustrates the five rules for one ledger step. For the definitions below, s denotes a ledger step, D_s its agreed decision, and S_s the proposal references selected by that decision. Validators must interpret D_s identically; additional local observations cannot change the canonical input.

3.1 Fixed proposal vectors

Selected object. Each step reserves a labeled position for each designated proposer, $D_s = (e_{s,1}, \ldots, e_{s,k})$ with $e_{s,i} \in \{p_{s,i}, \perp\}$. Here $p_{s,i}$ is proposer i's contribution and $\perp$ denotes an explicitly empty position. A related rule selects a prefix $D_s = (p_{s,\pi_s(1)}, \ldots, p_{s,\pi_s(r_s)})$, with $0 \leq r_s \leq k$, of a predetermined ranking π_s. The positions are recreated each step. Closing a ranked prefix can exclude its remaining positions without giving them the same meaning as explicit empty

entries. Cadence, BRAID, BigDipper-7, and Strong Prefix Consensus illustrate this family [8,87,113,112].
Missing proposals. The protocol must resolve the reserved positions by accepting proposals, using empty entries, closing a prefix, or invoking fallback. This makes a missing contribution attributable to a position, but one unresolved position can delay the whole step.
Recovery of selected bodies. Validators still need the data for every retained contribution. Agreement on a vector does not supply reveal, duplicate handling, or application ordering rules. Appendix B.6 details the failure and recovery behavior of the individual configurations.

3.2 Selected proposal collections

Selected object. The protocol selects from eligible candidates without reserving a position for each author. For candidate references $\mathcal{C}_s$, the output can be a set $D_s = Q_s \subseteq \mathcal{C}_s$ or an ordered list of distinct candidates. It can also be a collection of attestations whose support thresholds determine which candidates qualify. These are variants of the same membership rule. Whether selection also fixes order is recorded separately.

HoneyBadgerBFT selects an asynchronous common subset (ACS): validators agree on a sufficiently large common set of proposed batches without waiting for every proposer [71]. GNR selects through an attestation map; Multiplicity selects signed transaction bundles; and Raptr selects a supported prefix of a leader's ordered list of independently authored batches [41,38,107]. Constellation selects proposals by receipt thresholds over a rolling window. Candidates can survive across cycles, but selection does not advance lane prefixes or follow graph ancestry [5].
Missing proposals. The protocol can progress once enough candidates qualify. That threshold need not protect a particular candidate. For example, selecting $n - f$ proposals can still exclude a specified correct proposer's input.
Recovery of selected bodies. Every selected reference needs the recovery promised by its dissemination scheme. A common collection then needs a common execution order and rules for repeated or conflicting transactions. Compact agreement over references does not remove that work.

3.3 Persistent proposal streams

Selected object. A *lane* is a proposal history that continues across ledger steps. If one decision includes lane A through A_2 and lane B through B_1, the next can advance to A_3 while retaining B_1. In contrast, a fresh vector opens new positions for both authors at each step. For append-only lanes $L_i = (p_{i,1}, p_{i,2}, \ldots)$, a *cut* specifies how far each lane contributes, $D_s = (\ell_{s,1}, \ldots, \ell_{s,k})$ with $\ell_{s+1,i} \geq \ell_{s,i}$. The index $\ell_{s,i}$ identifies the selected prefix of lane i. Related designs merge outputs from continuing ordering instances. Autobahn, DispersedLedger, Sei Giga, and the multiple-lane instantiation of Hermes are representative configurations [42,114,68,85].

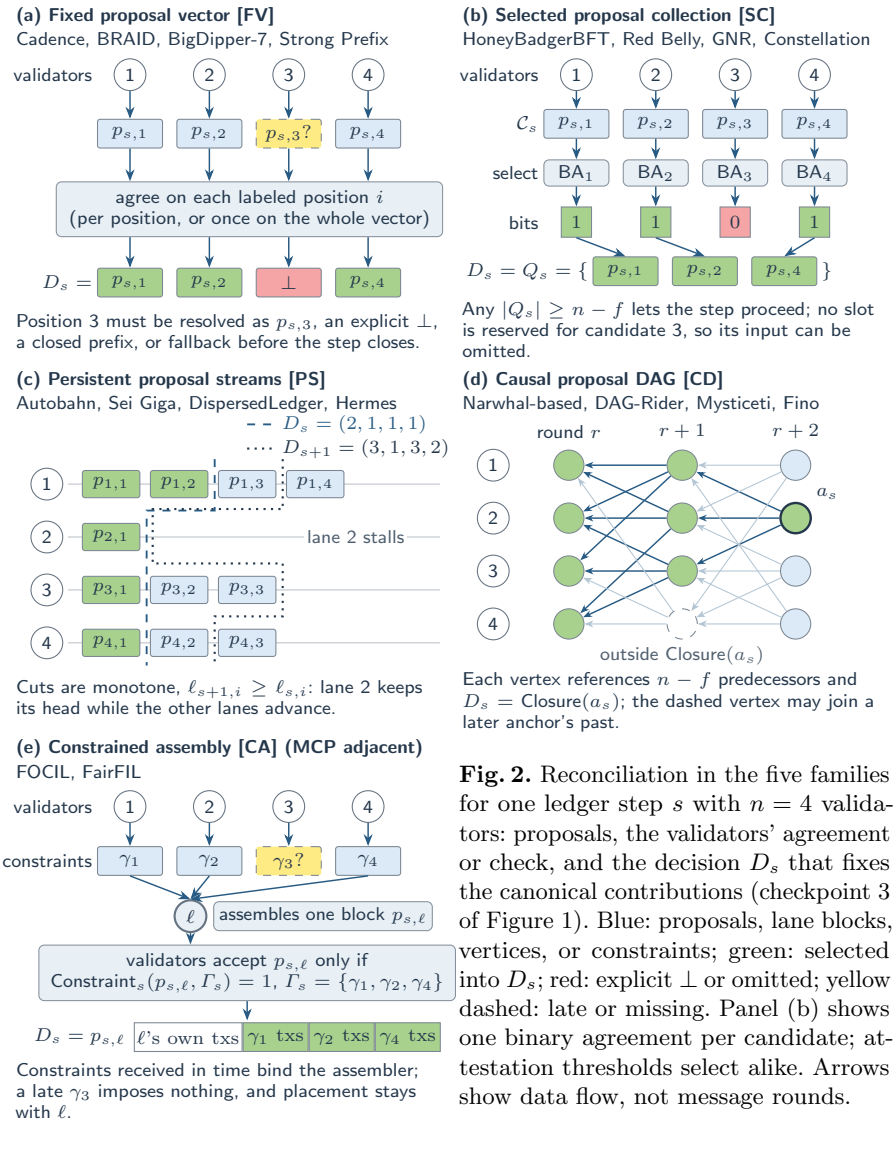

Fig. 2. Reconciliation in the five families for one ledger step s with $n = 4$ validators: proposals, the validators' agreement or check, and the decision D_s that fixes the canonical contributions (checkpoint 3 of Figure 1). Blue: proposals, lane blocks, vertices, or constraints; green: selected into D_s; red: explicit $\perp$ or omitted; yellow dashed: late or missing. Panel (b) shows one binary agreement per candidate; attestation thresholds select alike. Arrows show data flow, not message rounds.

Missing proposals. An inactive lane can retain its previous head while others advance. This isolates a slow path only if the merge rule can progress without that path's next item. Users assigned solely to the stalled path still need another route or a reassignment rule.

Recovery of selected bodies. A cut can select many accumulated blocks at once after a stall, and validators must still retrieve and execute them. Recovering consensus progress and clearing the execution backlog have different costs.

3.4 Causal proposal DAGs

Selected object. Proposals form a directed acyclic graph through references to earlier proposals, including those of other authors. In an anchor-based design, agreement selects a vertex a_s. Its *causal past* $\mathsf{Past}(a_s)$ contains the vertices reachable by following those references, and $\mathsf{Closure}(a_s) = \{a_s\} \cup \mathsf{Past}(a_s)$. The protocol selects eligible, previously unordered contributions from this closure and derives their order under its validity and certificate rules. Narwhal supplies the certified graph used by Narwhal–HotStuff and Bullshark; committing a reference brings its causal past into the ledger [31,100]. DAG-Rider, Mysticeti, and Fino also use causal DAG reconciliation [50,7,65].
Missing proposals. A vertex outside one selected closure may enter the past of a later anchor. Whether it does, and how soon, depends on the reference and commit rules. The graph shape alone gives no transaction-inclusion bound.
Recovery of selected bodies. Validators may need to retrieve ancestors before validating or processing a selected vertex. Certificates, retention rules, and traversal determine the recovery burden. A deterministic traversal gives a common order once its inputs are fixed; fairness requires a further criterion.

3.5 Constrained assembly

Selected object. One leader or builder assembles the output. Other participants supply constraints on that assembly, rather than independently selectable proposal bodies. Abstractly, $D_s = p_{s,\ell}$ with $\mathsf{Constraint}_s(p_{s,\ell}, \Gamma_s) = 1$, where ℓ is the assembler and Γ_s is the constraint evidence. FOCIL and FairFIL are examples [106,105,101]. Their list authors constrain a separately constructed block. In Multiplicity, by contrast, the signed bundles themselves become selected contributions, which places it in the selected-collection family [38].
Missing proposals. There is no reserved output position for each constraint contributor. Missing or late messages instead affect which constraints apply. Inclusion therefore depends on the list's eligibility, propagation, and capacity rules.
Recovery of selected bodies. The base block protocol supplies body delivery. Constraining omission leaves the assembler's other powers, such as access to public contents and placement within the block, to the rest of the configuration.

3.6 Comparison of reconciliation models

Table 1 compares the families across the ten properties; Appendix B gives the full notation. In the table, k counts reserved positions, candidate proposals, or lanes, not the validator count n, and q is a threshold on those k candidates: a three-of-four rule proceeds without one candidate but does not protect a transaction carried only by that candidate, whereas a four-position vector needs a decision for every required position. These are conditions on selection, not Byzantine fault bounds. The safety and ordering rows name work the protocol must perform,

Table 1. Properties across families. Colors distinguish structural benefits, conditions, and blocking points, not overall quality. k counts positions, candidates, or lanes in the relevant column; q is a candidate-selection threshold where used, not a validator quorum. Communication cells count logical metadata before compression; network costs require a concrete protocol.

Property	Fixed vector (FV) [8]	Selected collection (SC) [71]	Persistent streams (PS) [42]	Causal DAG (CD) [100]	Constrained assembly (CA) [105]
Selected object	k reserved positions	Eligible collection	Continuing prefixes	Causal closure	Constrained block
Safety	Agree on entries	Agree on collection	Prefix agreement + merge	Commit rule + closure	Base-protocol agreement
Liveness	1 unresolved of k can block an all-position path*	q-of-k selection permits $k - q$ absent candidates	1 stalled lane need not stop other prefixes*	Missing author: reference and commit rules decide	1 silent assembler can lose its turn; base-protocol recovery
Inclusion	Entry protection needs admission	A quorum can omit the target	Lane admission + service	Vertex must enter closure	List eligibility + capacity
Hiding	Not implied	Not implied	Not implied	Not implied	Not implied
Last-look protection	Bind body + entry choice	Bind body + selector	Bind heads + cut choice	Bind vertices + anchor choice	Bind assembler choices
Ordering / fairness	Merge entries	Order collection	Interleave prefixes	Traverse closure	Constrained assembler
Communication	k entries; payloads + proofs	Selected references; payloads + proofs	k heads/cut; continuing data	Anchors + ancestor evidence; data	List contents + base-protocol block
Rate / goodput	Duplicates + conflicts	Selection + recovery	Parallel input; execution backlog	Parallel gossip; dependencies	Assembler capacity
Latency	Slowest required entry or fallback*	Qualification + recovery	Cut + recovery + backlog	Commit + ancestor fetch	List cutoff + base-protocol path
Cadence	Vector per step	Collection per step	Independent producers; cuts	Rounds / anchors	Base-protocol slot + list schedule

Green: possible fault isolation Yellow: condition Red: blocking point

Blue: structure or work Gray: outside the family rule . *A fixed vector can use empty entries, prefix closure, or fallback; one faulty proposer does not universally stop ledger progress. Stream isolation requires a cut that can retain an old head. Protocol quorums and recovery still apply. Fairness and hiding require separate mechanisms.

and "Bind" means closing the indicated choice before the later information it should not exploit becomes available.

The same reconciliation structure can operate under different membership and finality assumptions: fixed committees, stake or work weighting, and probabilistic producer elections affect how proposal rights arise and what confirmation means. Appendix B.7 explains the boundary comparisons, including blockDAGs, multiple chains, and Filecoin tipsets, and Appendix C gives the full protocol placements and their historical relationships.

4 Protocol Comparison

Tables 2–3 compare 39 configurations against the six guarantee properties in Section 2.6; Tables 12 through 16 report their communication, rate, latency, and cadence with each source's units and endpoints. The class tags are FV (fixed

vectors), SC (selected collections), PS (persistent streams), CD (causal DAGs), and CA (constrained assembly). The additional tags adj., comp., and bound. mean adjacent mechanism, component, and boundary comparison.

Reading a guarantee. Cells describe properties at the same level, under the network models of Section 2.6. *Eventual* protects eligible inputs without a fixed inclusion deadline; rows sharing this label keep distinct fault thresholds and stronger service results, which the notes identify. Green marks an established guarantee under the row's assumptions, whether native or inherited. Yellow marks a narrower service, scope, or unresolved qualification; orange marks an economic argument. Thus eventual inclusion is green, while "include-or-empty" remains yellow: empty outputs may continue without serving the target. Blue gives a mechanism or delegated responsibility; gray identifies an unspecified mechanism or integration in the cited configuration. A bare *Base protocol* entry leaves the property to that protocol; it does not establish a new guarantee. Purple marks plaintext or demonstrated exposure. Blind selection or ordering protects that choice, not every later action or public signal. A last-look cell beginning with *No* identifies a choice that can still depend on information observed after another participant has fixed its input. The notes specify that choice and the binding point. An *unspecified* entry instead identifies missing protocol rules. Neither classification asserts that every reaction changes execution order or earns a profit.

Protocol names link to the guarantee notes in Appendix E; Table 4 maps every row to its guarantee and performance panels, and Appendix E.2 expands recurring phrases.

5 Guarantees and Tradeoffs

We now ask what the comparison implies for the service provided to a transaction, as distinct from progress of the ledger as a whole. The differences arise at admission and selection, access to information, ordering, and execution.

Inclusion under different receipt premises. A leader can keep omitting one proposal while selecting enough others for the ledger to advance. Even guaranteed proposal selection protects only transactions that the proposer admits. Recovery can then delay an included transaction, and execution can reject a conflict.

Conversely, FOCIL constrains a single assembler: its next-slot rule protects transactions already on eligible, non-equivocating lists delivered before the freeze deadlines, subject to validity and remaining gas. Receipt by a list author does not itself satisfy those premises [106,105]. AMP requires all correct validators to attest the protected proposal; Cadence protects timely correct proposals after its synchrony grace period; and GNR protects timely submissions in nonempty slots while permitting empty runs [21,8,41]. These are different services, even when every construction makes progress.

Table 2. Protocol comparison (1/2): the six guarantee properties; communication, rate, latency, and cadence for every row are in Tables 12 through 16. Names link to expanded notes. Notation: Table 5; row-by-row references: Table 4.

Protocol / class	Safety	Liveness	Inclusion	Hiding	Last-look protection	Ordering / fairness
Multiplicity [38] [SC]	Base protocol	Base protocol	Tx in $> 1/3$ bundles required	Plaintext	No; bundle choice	Bundle selection
FOCIL [105] [CA; adj.]	Base protocol	Base protocol	Eligible listed tx: next slot	Plaintext	No; assembly	Assembler order
BRAID [87] [FV]	LMD-GHOST + finality	Base protocol	Bribe all k; economic model	Sealing unspecified	Release unspecified	Union, dedup, fee order
MBA [88] [FV]	Base protocol	Base protocol	Entry validity / admission	Optional overlay	Overlay dependent	Vector; merge unspecified
Cadence [8] [FV]	$3f < n$	PSync	Timely correct proposal: same slot	Until slot deadline	New bodies only	Proposal vector
GNR + coded hiding [41] [SC]	$5f < n$	PSync	Include-or-empty	Other nodes; until inevitable inclusion	Public-signal look remains	Select, reveal, order
Supernova Part One [88] [SC]	Integration unspecified	Recovery unspecified	Receipt rule	Hiding unspecified	Binding rule unspecified	Receipt selection; merge unspecified
Constellation [5] [SC]	$< 20\%$ Byzantine stake	PSync	Include-or-empty; λ-cycle window	Plaintext	No; proposals	Threshold selection
AUCIL [109] [CA; adj.]	Base protocol	Base protocol	Economic incentives	Plaintext	No; assembly	Assembler order
BigDipper-7 [113] [FV]	$3f < n$	PSync	Certified data or no block	Plaintext	No; residual selection	Upper layer
AMP [21] [SC]	$3f < n$	PSync	All correct attest: next height	Plaintext	No; proposals	Deterministic payload merge
Ethereum (two proposers) [78] [FV]	Base protocol	Base protocol	$2\times$ bribe cost; model	Plaintext	No; payloads	Pair, dedup, tip split
HoneyBadgerBFT [71] [SC]	$3f < n$	Async	Backlog-bound epochs	Until selection	Blind selection	ACS + merge
Dumbo2 (ACS) [44,43] [SC]	$3f < n$	Async	Subset size only	Plaintext	No; ACS inputs	ACS; upper-layer order
Red Belly [28] [SC]	$3f < n$	PSync	Eventual; conflict-free input	Plaintext	No; proposals	Set; application order
Gatling (Simplex) [92] [PS]	$3f < n$	PSync	Eventual; all correct input	Plaintext	No; proposals	Interleaved instances
Mir-BFT [102] [PS]	$3f < n$	PSync	Eventual	Plaintext	No; proposals	Parallel batches
ISS [103] [PS]	$3f < n$	PSync	Eventual	Plaintext	No; proposals	Segment merge
Ladon [63,92] [PS]	Disputed	Claimed PSync	Eventual claimed; order disputed	Plaintext	No; proposals	Rank merge; disputed

Guarantee | Restricted scope | Economic claim | Plaintext / exposure | Disputed / refuted | Mechanism / dependency | Unspecified

PSync: partial synchrony. Eventual: no fixed inclusion deadline. No: the named choice remains reactive. All entries retain the linked assumptions.

Table 3. Protocol comparison (2/2): the six guarantee properties; communication, rate, latency, and cadence for every row are in Tables 12 through 16. Names link to expanded notes. Notation: Table 5; row-by-row references: Table 4.

Protocol / class	Safety	Liveness	Inclusion	Hiding	Last-look protection	Ordering / fairness
Orthrus [62] [PS]	$3f < n$; objects	PSync	Eventual; may abort	Plaintext	No; proposals	Partial / global order
Raptr [107] [SC]	$3f < n$	PSync	Eventual	Plaintext	No; leader list	List prefix
Strong Prefix [112] [FV]	$3f < n$	PSync	Honest input: at most f omissions	Plaintext	No; inputs	Ranked vector prefix
DispersedLedger [114] [PS]	$3f < n$	Async	Eventual	Plaintext	No; inputs	Linked prefixes
Autobahn [42] [PS]	$3f < n$	PSync	Eventual	Plaintext	No; fresh cuts	Cut + merge
Sei Giga [68] [PS]	$3f < n$	PSync	Eventual	Sedna proposed; bounded leakage	Proposed; public look remains	Cut + merge
Hermes (v2, Autobahn lanes) [85] [PS]	$5f < n$	PSync	Eventual	Plaintext	No; fresh cuts / skips	Common lane prefix
Multimmit (v4) [59] [PS]	$5f < n$	PSync	Eventual	Plaintext	No; lane inputs	Certified lane interleaving
Narwhal–HotStuff [31] [CD]	$3f < n$	PSync	Eventual	Plaintext	No; vertices	Causal past
Tusk [31,93] [CD]	$3f < n$	Async claim refuted	Fails in analyzed attack	Plaintext	No; vertices	Causal DAG order
Bullshark (PSync) [100] [CD]	$3f < n$	PSync	Eventual	Plaintext	No; vertices	Causal DAG order
DAG-Rider [50] [CD]	$3f < n$	Async	Eventual	Plaintext	No; vertices	Causal DAG order
Mysticeti-C [7] [CD]	$3f < n$	PSync	Eventual	Plaintext	No; vertices	DAG traversal
Fino [65] [CD]	$3f < n$	PSync	Eventual	Until ordering	Blind ordering	Order, then decrypt
SPECTRE [97] [CD; bound.]	$\alpha < 1/2$; pairs	Sync	Published nonconflicting tx: eventual	Plaintext	No; mining inputs	Pairwise precedence
GHOSTDAG [98] [CD; bound.]	$\alpha < (1-\eta)/2$	Sync	Published blocks: probabilistic	Plaintext	No; mining inputs	DAG total order
Conflux [60] [CD; bound.]	$\alpha < 1/2$	Sync	Published blocks: probabilistic	Plaintext	No; mining inputs	Pivot-chain DAG order
Prism [9] [CD; bound.]	$\alpha < 1/2$	Sync	Tx in a block: probabilistic	Plaintext	No; mining inputs	Proposer / voter DAGs
OHIE [116] [PS; bound.]	$\alpha < 1/2$	Sync	All honest receive: probabilistic bound	Plaintext	No; mining inputs	Ranked chain prefixes
Filecoin EC [81,?] [SC; bound.]	$\beta < \beta_\star$	Sync + beacon	All honest receive: probabilistic bound	Plaintext	No; block inputs	Tipset chain

Guarantee | Restricted scope | Economic claim | Plaintext / exposure | Disputed / refuted | Mechanism / dependency | Unspecified

PSync: partial synchrony. Eventual: no fixed inclusion deadline. No: the named choice remains reactive. All entries retain the linked assumptions.

Hiding and last look. GNR hides a transaction from other protocol nodes, but not its initial receiver. Cadence restricts new reactive proposals while leaving selection among previously eligible faulty candidates possible [41,8]. If A commits at c_A and B can act until $c_B > c_A$, a public price change between those times can still benefit B. Last-look protection must name the choice that closes. Giga proposes Sedna to restrict access to transaction contents before reconstruction; its content protection and residual exposure to public signals are therefore recorded separately [68,86].

Ordering and participant influence. A deterministic merge fixes the order once its inputs are fixed. Earlier, participants may influence selection, delivery, lane assignment, fees, timestamps, dependencies, or tie-breaking. Fair-ordering protocols constrain some of these choices using participants' observations, but their guarantees depend on eligible inputs and the treatment of cycles and batches [52,19,51,108]. An MCP fairness claim needs an eligibility rule for recovered transactions and a rule turning any permitted batch into an execution sequence.

Ordering throughput and executed goodput. Repeated copies consume bandwidth; distinct transactions can compete for the same nonce, balance, or liquidity. Missing bodies and delayed reveal add work before execution. Faster proposal production or ordering therefore need not increase distinct successful state transitions. Sei Giga illustrates the endpoint distinction: it reports executed gas and ordering latency, with state attestation later in the pipeline [68].

6 Failure Recovery and Execution

The guarantees in Section 5 depend on the work between agreement and execution. A missing proposal can delay selection; a selected but unavailable body can delay execution after selection. The reconciliation rule sets how far each failure spreads.

6.1 Effects of a missing contribution

In a fixed vector, resolving one position can affect the whole step. In Cadence, a split that prevents a quorum for one proposer can trigger slot-wide fallback, while a fully silent proposer need not; later slots may decide while ledger-prefix delivery and stateful execution wait for the unresolved slot [8]. Collections omit unqualified candidates. Streams can retain an inactive lane's head while others advance, as Autobahn does during recovery, although a client assigned only to the stalled lane still needs another route; DispersedLedger delivers correct authors' omitted blocks through later decisions [42,114]. A causal DAG can include a late vertex through a later decision at the cost of successive ancestor fetches, and a late or ineligible inclusion list may impose no obligation on the assembled block [105]. Appendix B.6 records the configuration-specific rules and an illustrative model of how fallback risk grows with the number of proposers.

6.2 Recovering and revealing selected bodies

After a contribution is selected, validators need its body. Narwhal certifies disseminated data before ordering; Autobahn can finalize a cut while retrieval proceeds outside the consensus critical path [31,42]. Under Autobahn's static fault model, an $f+1$-signature availability certificate guarantees a correct holder, and Lemma 10 guarantees retrieval of every committed cut and its referenced lane prefixes under the protocol's retention and serving rules. A recovery deadline additionally depends on network and resource assumptions.

Hidden contents add an opening step. HoneyBadgerBFT decrypts after batch selection; Fino orders before threshold opening [71,65]. Withheld shares, malformed ciphertexts, or failed reconstruction can therefore delay execution after the membership or ordering decision. A complete configuration specifies the treatment of each failure and preserves the protected choice until the appropriate reveal event. Appendix D and Table 17 cover the encryption and fair-ordering alternatives.

Committee and key handoff. Membership changes must preserve obligations for certified but unexecuted contributions. The successor configuration needs an authenticated account of the canonical history, agreed lane positions, and pending bodies. Preserving those commitments is a safety requirement; establishing continued access to the bodies is a separate liveness requirement. Retiring holders must retain and serve pending data until the specified transfer of custody or completion condition is satisfied. Transferring certificates alone does not transfer the data they certify. Encrypted contributions additionally require an opening path that survives key rotation. The handoff must preserve access to sufficient old shares, or specify a secure transition that retains the ability to decrypt pending ciphertexts without revealing their contents before the protected choice closes. Recovery, retention, and secrecy claims must use a corruption model that covers the transition itself. A fault bound stated separately for each epoch does not by itself establish availability or confidentiality across successive committees. These are composition obligations for any claimed service that spans the transition.

6.3 Execution backlog and measured performance

Restoring agreement does not finish recovery. One cut can make many accumulated lane blocks canonical at once, but validators must still obtain their bodies, remove duplicates, resolve transaction dependencies, and execute the sequence. The analogous DAG path includes recovery of the required ancestors. Progress at the ordering layer can coexist with a growing execution backlog.

Ordering latency, execution readiness, executed goodput, and state attestation therefore capture different parts of recovery, and results depend on offered load, duplicate and conflicting work, withholding, and the capacity reserved for recovery traffic. Reporting the backlog and the time to clear it exposes costs that a steady-state throughput figure can miss; the notes in Appendix E.5 preserve each source's measured endpoint and fault setting.

7 Fees and Proposer Independence

Recovery, duplicate dissemination, and execution all consume resources. The same transaction can create work for several proposers while executing once. Fees determine who bears that cost, and shared builders or other operators determine how independent the proposal paths really are.

7.1 Paying for concurrent work

A fee rule must identify the paid service: admission, bytes disseminated, a selected contribution, or execution. It must also allocate rewards among carriers and specify how copies consume capacity. Leaderless fee mechanisms make these choices explicit [40]. Rewarding several carriers can increase the cost of censorship while encouraging duplicate traffic; penalizing copies can reduce a user's alternative routes [91]. Fees for parallel execution address contention and contingent execution, but do not by themselves settle payment for redundant proposals, withheld data, or failed reveal [76,1,110].

Economic inclusion claims depend on the allowed deviations and bribes. For example, the FOCIL fee analysis restricts incentives, bribes, and demand in the specified ways listed in Appendix B.6 [104]. Fox, Pai, and Resnick analyze suppression over a fixed window [37]. Such results retain their specified participants and horizon; they do not automatically price a coalition that also controls relays, data holders, or decryption shares.

7.2 Admission solvency and capital efficiency

Several lanes can accept work against the same unsettled balance, so charging only the transaction that eventually executes can leave duplicate or failed work unpaid. A design promising payment for all accepted work needs prepayment, enforceable reservations, or an explicit credit and loss model. Sedna, for example, requires prepaid byte fees at admission; its admission abstraction does not itself implement a shared reservation ledger across lanes [86]. If user u's unsettled charges at each admissible lane $i \in S_u$ are capped at $Q_{u,i}$ and can become payable together, a reserved balance $R_u \geq \sum_{i \in S_u} Q_{u,i}$ suffices. This accounting condition, not a Sedna theorem, is developed with its assumptions in Appendix F.

7.3 Shared builders and other intermediaries

Proposer-builder separation delegates construction and ordinarily produces one builder-supplied payload [77]. In an MCP system, several proposers may obtain their bodies from the same builder. One relay can similarly dominate ingress, and one operator can hold several decryption shares. Studies of private order flow and builder integration show how information and rapid access can concentrate construction markets [46,79,10]. Enshrined PBS changes fair-exchange and availability assumptions while leaving this concentration possible [30,56].

Concurrency also changes the value of particular actions. Copying within a step, auctions between proposers, and races to form availability proofs have been analyzed for a Sei Giga-style pipeline [67]. A fragment that completes reconstruction or a share that enables decryption can have more bargaining value than an earlier redundant contribution [69]. Fee and censorship analyses therefore need to account for overlapping operators, side payments, and control over admission, selection, storage, and reveal.

8 Conclusion

Multiple concurrent proposers give users several routes into one ledger update. The four direct MCP families differ in how those contributions become canonical; constrained assembly instead constrains one assembled block. The comparison shows that these choices determine where omission, delay, and duplicate work can arise, while hiding and fair ordering require their own mechanisms.

For the auction bid in the introduction, success depends on admission, selection, recovery, order, execution, and the timing of competing choices. A faster proposal rate or more proposer identities establishes only part of that service. Explicit inclusion premises, recovery rules, payment obligations, and control assumptions make each configuration's benefits precise and its limitations visible.

References

1. Acilan, B., Constantinescu, A., Heimbach, L., Wattenhofer, R.: Transaction fee market design for parallel execution. In: 7th Conference on Advances in Financial Technologies. Leibniz International Proceedings in Informatics (LIPIcs), vol. 354, pp. 23:1–23:25 (2025). https://doi.org/10.4230/LIPIcs.AFT.2025.23
2. Alpos, O., David, B., Kamarinakis, N., Zindros, D.: An overview of the efficiency and censorship-resistance guarantees of widely-used consensus protocols (2025). https://doi.org/10.48550/arXiv.2504.03588, https://arxiv.org/abs/2504.03588v2, version 2, 18 August 2025
3. Alqahtani, S., Demirbas, M.: BigBFT: A multileader byzantine fault tolerance protocol for high throughput. In: 2021 IEEE 40th International Performance Computing and Communications Conference (IPCCC). pp. 1–10. IEEE (2021). https://doi.org/10.1109/IPCCC51483.2021.9679432
4. Antoniadis, K., Benhaim, J., Desjardins, A., Poroma, E., Gramoli, V., Guerraoui, R., Voron, G., Zablotchi, I.: Leaderless consensus. Journal of Parallel and Distributed Computing **176**, 95–113 (2023). https://doi.org/10.1016/j.jpdc.2023.01.009
5. Anza: Constellation (MCP) on Solana. Protocol design documentation (2026), https://constellation.anza.xyz/, living document consulted 8 September 2026
6. Avarikioti, Z., Heimbach, L., Schmid, R., Vanbever, L., Wattenhofer, R., Wintermeyer, P.: FnF-BFT: Exploring performance limits of BFT protocols (2020). https://doi.org/10.48550/arXiv.2009.02235

7. Babel, K., Chursin, A., Danezis, G., Kichidis, A., Kokoris-Kogias, L., Koshy, A., Sonnino, A., Tian, M.: Mysticeti: Reaching the latency limits with uncertified DAGs. In: Network and Distributed System Security Symposium (2025). https://doi.org/10.14722/ndss.2025.240929, https://www.ndss-symposium.org/ndss-paper/mysticeti-reaching-the-latency-limits-with-uncertified-dags/, the cited preprint first appeared in 2023
8. Babel, K., Elsheimy, F., Heimbach, L., Jalalzai, M.M., Klenze, T., Komatovic, J., Milionis, J., Setrin, M., Shoup, V.: Cadence: Extreme pipelining with multiple concurrent proposers (2026), https://arxiv.org/abs/2607.02275v2, version 2, 7 July 2026
9. Bagaria, V., Kannan, S., Tse, D., Fanti, G., Viswanath, P.: Prism: Deconstructing the blockchain to approach physical limits. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 585–602 (2019). https://doi.org/10.1145/3319535.3363213, https://arxiv.org/abs/1810.08092
10. Bahrani, M., Garimidi, P., Roughgarden, T.: Centralization in block building and proposer-builder separation. In: Clark, J., Shi, E. (eds.) Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 14744, pp. 331–349. Springer (2025). https://doi.org/10.1007/978-3-031-78676-1_19, https://arxiv.org/abs/2401.12120, proceedings of FC 2024
11. Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., Danezis, G.: SoK: Consensus in the age of blockchains. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies. pp. 183–198 (2019). https://doi.org/10.1145/3318041.3355458
12. Baum, C., Chiang, J.H.y., David, B., Frederiksen, T.K., Gentile, L.: SoK: Mitigation of front-running in decentralized finance. In: Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 13412, pp. 250–271 (2023). https://doi.org/10.1007/978-3-031-32415-4_17, https://eprint.iacr.org/2021/1628
13. Bebel, J., Ojha, D.: Ferveo: Threshold decryption for mempool privacy in BFT networks. Cryptology ePrint Archive, Paper 2022/898 (2022), https://eprint.iacr.org/2022/898
14. Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing. pp. 52–61 (1993). https://doi.org/10.1145/167088.167109
15. Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience. In: Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing. pp. 183–192 (1994). https://doi.org/10.1145/197917.198088
16. Borran, F., Schiper, A.: A leader-free Byzantine consensus algorithm. In: Distributed Computing and Networking. Lecture Notes in Computer Science, vol. 5935, pp. 67–78 (2010). https://doi.org/10.1007/978-3-642-11322-2_11
17. Buterin, V.: Possible futures of the Ethereum protocol, part 3: The Scourge. Author's website (2024), https://vitalik.eth.limo/general/2024/10/20/futures3.html
18. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. In: Advances in Cryptology – CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139, pp. 524–541 (2001). https://doi.org/10.1007/3-540-44647-8_31

19. Cachin, C., Mićić, J., Steinhauer, N., Zanolini, L.: Quick Order Fairness. In: Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 13411, pp. 316–333 (2022). https://doi.org/10.1007/978-3-031-18283-9_15
20. Cachin, C., Tessaro, S.: Asynchronous verifiable information dispersal. In: 24th IEEE Symposium on Reliable Distributed Systems. pp. 191–201 (2005). https://doi.org/10.1109/RELDIS.2005.9, https://ieeexplore.ieee.org/document/1541196
21. Cason, D., Liao, G., Mena, S., Milošević, N., Seredinschi, A., Sforzin, A., Sousa, J., Vos, P.V.: AMP: Arc multi-proposer protocol with bounded inclusion guarantees (2026), https://arxiv.org/abs/2605.23677v1, version 1, 22 May 2026
22. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation. pp. 173–186. USENIX Association (1999), https://www.usenix.org/conference/osdi-99/presentation/practical-byzantine-fault-tolerance
23. Chan, B.Y., Pass, R.: Simplex consensus: A simple and fast consensus protocol. In: Theory of Cryptography. Lecture Notes in Computer Science, vol. 14372, pp. 452–479. Springer (2023). https://doi.org/10.1007/978-3-031-48624-1_17, https://eprint.iacr.org/2023/463
24. Choudhuri, A.R., Garg, S., Piet, J., Policharla, G.V.: Mempool privacy via batched threshold encryption: Attacks and defenses. In: 33rd USENIX Security Symposium. pp. 3513–3529 (2024), https://www.usenix.org/conference/usenixsecurity24/presentation/choudhuri
25. Chung, H., Shi, E.: Foundations of transaction fee mechanism design. In: Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms. pp. 3856–3899 (2023). https://doi.org/10.1137/1.9781611977554.ch150
26. Correia, M., Neves, N.F., Veríssimo, P.: From consensus to atomic broadcast: Time-free Byzantine-resistant protocols without signatures. The Computer Journal **49**(1), 82–96 (2006). https://doi.org/10.1093/comjnl/bxh145
27. Crain, T., Gramoli, V., Larrea, M., Raynal, M.: DBFT: Efficient leaderless Byzantine consensus and its application to blockchains. In: 2018 IEEE 17th International Symposium on Network Computing and Applications. pp. 1–8 (2018). https://doi.org/10.1109/NCA.2018.8548057, https://ieeexplore.ieee.org/document/8548057
28. Crain, T., Natoli, C., Gramoli, V.: Red Belly: A secure, fair and scalable open blockchain. In: 2021 IEEE Symposium on Security and Privacy. pp. 466–483 (2021). https://doi.org/10.1109/SP40001.2021.00087
29. Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., Juels, A.: Flash Boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In: 2020 IEEE Symposium on Security and Privacy. pp. 910–927 (2020). https://doi.org/10.1109/SP40000.2020.00040
30. D'Amato, F., Flaig, N., Monnot, B., Neuder, M., Potuz, Traglia, J., Tsao, T.: EIP-7732: Enshrined proposer-builder separation. Ethereum Improvement Proposal (2024), https://eips.ethereum.org/EIPS/eip-7732, draft specification consulted 15 August 2026
31. Danezis, G., Kokoris-Kogias, E., Sonnino, A., Spiegelman, A.: Narwhal and Tusk: A DAG-based mempool and efficient BFT consensus. In: Proceedings of the Seventeenth European Conference on Computer Systems. pp. 34–50 (2022). https://doi.org/10.1145/3492321.3519594, https://arxiv.org/abs/2105.11827
32. Döttling, N., Hanzlik, L., Magri, B., Wohnig, S.: McFly: Verifiable encryption to the future made practical. In: Financial Cryptography and Data Security.

Lecture Notes in Computer Science, vol. 13950, pp. 252–269 (2024). https://doi.org/10.1007/978-3-031-47754-6_15, fC 2023 revised selected papers; the preprint first appeared in 2022
33. Doudou, A., Schiper, A.: Muteness detectors for consensus with Byzantine processes. In: Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing. p. 315 (1998). https://doi.org/10.1145/277697.277772
34. Duan, S., Reiter, M.K., Zhang, H.: BEAT: Asynchronous BFT made practical. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 2028–2041 (2018). https://doi.org/10.1145/3243734.3243812
35. Duan, S., Wang, X., Zhang, H.: FIN: Practical signature-free asynchronous common subset in constant time. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 815–829 (2023). https://doi.org/10.1145/3576915.3616633, https://eprint.iacr.org/2023/154
36. Elsheimy, F., Kaklamanis, I., Wadhwa, S., Papamanthou, C., Zhang, F.: Censorship resistance vs throughput in multi-proposer BFT protocols. Cryptology ePrint Archive, Paper 2026/126 (2026), https://eprint.iacr.org/2026/126
37. Fox, E., Pai, M.M., Resnick, M.: Censorship resistance in on-chain auctions. In: 5th Conference on Advances in Financial Technologies. Leibniz International Proceedings in Informatics, vol. 282, pp. 19:1–19:20 (2023). https://doi.org/10.4230/LIPIcs.AFT.2023.19
38. Fox, E., Resnick, M.: Multiplicity: A gadget for multiple concurrent block proposers. Ethereum Research forum post (2023), https://ethresear.ch/t/multiplicity-a-gadget-for-multiple-concurrent-block-proposers/14962, posted 3 March 2023
39. Gao, Y., Lu, Y., Lu, Z., Tang, Q., Xu, J., Zhang, Z.: Dumbo-NG: Fast asynchronous BFT consensus with throughput-oblivious latency. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 1187–1201. ACM (2022). https://doi.org/10.1145/3548606.3559379
40. Garimidi, P., Heimbach, L., Roughgarden, T.: Transaction fee mechanism design for leaderless blockchain protocols (2025), https://arxiv.org/abs/2505.17885
41. Garimidi, P., Neu, J., Resnick, M.: Multiple concurrent proposers: Why and how (2025), https://arxiv.org/abs/2509.23984v1, version 1, 28 September 2025; equivalent record at Cryptology ePrint 2025/1772
42. Giridharan, N., Suri-Payer, F., Abraham, I., Alvisi, L., Crooks, N.: Autobahn: Seamless high speed BFT. In: Proceedings of the 30th Symposium on Operating Systems Principles. pp. 1–23 (2024). https://doi.org/10.1145/3694715.3695942, https://arxiv.org/abs/2401.10369v4, version 4, 20 January 2025
43. Guo, B., Lu, Y., Lu, Z., Tang, Q., Xu, J., Zhang, Z.: Speeding Dumbo: Pushing asynchronous BFT closer to practice. In: Network and Distributed System Security Symposium (2022). https://doi.org/10.14722/ndss.2022.24385, https://www.ndss-symposium.org/wp-content/uploads/2022-385-paper.pdf
44. Guo, B., Lu, Z., Tang, Q., Xu, J., Zhang, Z.: Dumbo: Faster asynchronous BFT protocols. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 803–818 (2020). https://doi.org/10.1145/3372297.3417262
45. Gupta, S., Hellings, J., Sadoghi, M.: RCC: Resilient concurrent consensus for high-throughput secure transaction processing. In: 2021 IEEE 37th International Conference on Data Engineering. pp. 1392–1403 (2021). https://doi.org/10.1

109/ICDE51399.2021.00124, https://arxiv.org/abs/1911.00837v2, version 2, 2 November 2020
46. Gupta, T., Pai, M.M., Resnick, M.: The centralizing effects of private order flow on proposer-builder separation. In: 5th Conference on Advances in Financial Technologies. Leibniz International Proceedings in Informatics, vol. 282, pp. 20:1–20:15 (2023). https://doi.org/10.4230/LIPIcs.AFT.2023.20
47. Heimbach, L., Kiffer, L., Torres, C.F., Wattenhofer, R.: Ethereum's proposer-builder separation: Promises and realities. In: Proceedings of the 2023 ACM Internet Measurement Conference. pp. 406–420 (2023). https://doi.org/10.1145/3618257.3624824
48. Jovanovic, P., Kokoris-Kogias, L., Kumara, B., Sonnino, A., Tennage, P., Zablotchi, I.: Mahi-Mahi: Low-latency asynchronous BFT DAG-based consensus. In: 2025 IEEE 45th International Conference on Distributed Computing Systems (ICDCS). pp. 549–559. IEEE (2025). https://doi.org/10.1109/ICDCS63083.2025.00060
49. Kang, D., Chen, J., Dinh, T.T.A., Sadoghi, M.: FairDAG: Consensus fairness over multi-proposer causal design. Proceedings of the VLDB Endowment **19**(2), 265–278 (2025). https://doi.org/10.14778/3773749.3773763
50. Keidar, I., Kokoris-Kogias, E., Naor, O., Spiegelman, A.: All you need is DAG. In: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing. pp. 165–175 (2021). https://doi.org/10.1145/3465084.3467905
51. Kelkar, M., Deb, S., Long, S., Juels, A., Kannan, S.: Themis: Fast, strong order-fairness in Byzantine consensus. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 475–489 (2023). https://doi.org/10.1145/3576915.3616658
52. Kelkar, M., Zhang, F., Goldfeder, S., Juels, A.: Order-fairness for Byzantine consensus. In: Advances in Cryptology – CRYPTO 2020, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 451–480 (2020). https://doi.org/10.1007/978-3-030-56877-1_16
53. Khajehpour, A., Akbarinodehi, H., Jahanara, M., Feng, C.: A mempool encryption scheme for Ethereum via multiparty delay encryption. Cryptology ePrint Archive, Paper 2023/1612 (2023), https://eprint.iacr.org/2023/1612
54. Kniep, Q., Sliwinski, K., Wattenhofer, R.: SIMD-0326: Alpenglow. Solana Improvement Documents (2025), https://github.com/solana-foundation/solana-improvement-documents/blob/main/proposals/0326-alpenglow.md, created 25 July 2025; consulted 8 September 2026
55. Kniep, Q., Sliwinski, K., Wattenhofer, R.: Solana Alpenglow Consensus: Increased bandwidth, reduced latency. White paper, version 1.1 (2025), https://www.anza.xyz/alpenglow-1-1, version 1.1
56. Koegler, M.: SoK: Current state of Ethereum's enshrined proposer builder separation (2025), https://arxiv.org/abs/2506.18189
57. Korkmaz, K., Bruneau-Queyreix, J., Mokhtar, S.B., Réveillère, L.: ALDER: Unlocking blockchain performance by multiplexing consensus protocols. In: 2022 IEEE 21st International Symposium on Network Computing and Applications (NCA). pp. 9–18. IEEE (2022). https://doi.org/10.1109/NCA57778.2022.10013556
58. Kursawe, K.: Wendy, the good little fairness widget: Achieving order fairness for blockchains. In: Proceedings of the 2nd ACM Conference on Advances in Financial Technologies. pp. 25–36 (2020). https://doi.org/10.1145/3419614.3423263
59. Lewis-Pye, A., O'Grady, P.: Multimmit: Extending blocks for faster finality (2026), https://arxiv.org/abs/2607.21021v4, version 4, 28 August 2026

60. Li, C., Li, P., Zhou, D., Yang, Z., Wu, M., Yang, G., Xu, W., Long, F., Yao, A.C.C.: A decentralized blockchain with high throughput and fast confirmation. In: 2020 USENIX Annual Technical Conference. pp. 515–528 (2020), https://www.usenix.org/conference/atc20/presentation/li-chenxing
61. Liu, C., Duan, S., Zhang, H.: MiB: Asynchronous BFT with more replicas. IEEE Transactions on Dependable and Secure Computing **22**(5), 5508–5524 (2025). https://doi.org/10.1109/TDSC.2025.3568460, the cited preprint first appeared in 2021
62. Lyu, H., Xie, S., Niu, J., Beschastnikh, I., Zhang, Y., Sadoghi, M., Feng, C.: Orthrus: Accelerating multi-BFT consensus through concurrent partial ordering of transactions. In: 2025 IEEE 41st International Conference on Data Engineering. pp. 2615–2627 (2025). https://doi.org/10.1109/ICDE65448.2025.00197, https://arxiv.org/abs/2501.14732v3, extended version 3, 2 March 2025
63. Lyu, H., Xie, S., Niu, J., Feng, C., Zhang, Y., Beschastnikh, I.: Ladon: High-performance multi-BFT consensus via dynamic global ordering. In: Proceedings of the Twentieth European Conference on Computer Systems. pp. 226–242 (2025). https://doi.org/10.1145/3689031.3696102, https://arxiv.org/abs/2409.10954v2, extended version 2, 2 March 2025
64. Malkhi, D., Stathakopoulou, C., Yin, M.: BBCA-CHAIN: Low latency, high throughput BFT consensus on a DAG (2023). https://doi.org/10.48550/arXiv.2310.06335
65. Malkhi, D., Szalachowski, P.: Maximal extractable value (MEV) protection on a DAG. In: 4th International Conference on Blockchain Economics, Security and Protocols. Open Access Series in Informatics, vol. 110, pp. 6:1–6:17 (2023). https://doi.org/10.4230/OASIcs.Tokenomics.2022.6, https://arxiv.org/abs/2208.00940, tokenomics 2022 conference; proceedings published in 2023
66. Mao, Y., Junqueira, F.P., Marzullo, K.: Mencius: Building efficient replicated state machines for WANs. In: 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI 08). pp. 369–384. USENIX Association (2008), https://www.usenix.org/conference/osdi-08/mencius-building-efficient-replicated-state-machines-wans
67. Marsh, B., Landers, S.: MEV in multiple concurrent proposer blockchains. Frontiers in Blockchain **9**, 1782707 (2026). https://doi.org/10.3389/fbloc.2026.1782707
68. Marsh, B., Landers, S., Jog, J.: Sei Giga (2025), https://arxiv.org/abs/2505.14914v3, version 3, 29 June 2026
69. Marsh, B., Ranchal-Pedrosa, A.: A mechanism design overview of Sedna (2026), https://arxiv.org/abs/2603.17614v1, version 1, 18 March 2026; PIVOT-K is a mechanism defined in this paper
70. Martino, W., Quaintance, M., Popejoy, S.: Chainweb: A proof-of-work parallel-chain architecture for massive throughput. White paper (2018), https://d31d887a-c1e0-47c2-aa51-c69f9f998b07.filesusr.com/ugd/86a16f_029c9991469e4565a7c334dd716345f4.pdf, draft v15; consulted 8 September 2026
71. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The Honey Badger of BFT protocols. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 31–42 (2016). https://doi.org/10.1145/2976749.2978399, https://eprint.iacr.org/2016/199
72. Mirzaei, I., Amiri, M.J.: Fair on the surface: Transaction-ordering bias and MEV in Mysticeti DAG-based BFT protocol (2026), https://arxiv.org/abs/2607.13378v1, version 1, 15 July 2026

73. Moallemi, C.C., Pai, M.M., Robinson, D.: Latency advantages in common-value auctions (2025), https://arxiv.org/abs/2504.02077v2, version 2, revised May 2026
74. Momeni, P., Gorbunov, S., Zhang, B.: FairBlock: Preventing blockchain front-running with minimal overheads. In: SecureComm 2022. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 462, pp. 250–271 (2022). https://doi.org/10.1007/978-3-031-25538-0_14, conference held in 2022; proceedings volume published in 2023
75. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), https://bitcoin.org/bitcoin.pdf
76. Ndiaye, A.: Parallel execution fee mechanisms (2025), https://arxiv.org/abs/2410.09555v2, version 2, 24 February 2025
77. Neuder, M., Drake, J.: Why enshrine proposer-builder separation? a viable path to ePBS. Ethereum Research forum post (2023), https://ethresear.ch/t/why-enshrine-proposer-builder-separation-a-viable-path-to-epbs/15710
78. Neuder, M., Resnick, M.: Concurrent block proposers in Ethereum. Ethereum Research forum post (2024), https://ethresear.ch/t/concurrent-block-proposers-in-ethereum/18777, posted 23 February 2024
79. Pai, M.M., Resnick, M.: Structural advantages for integrated builders in MEV-Boost (2023), https://arxiv.org/abs/2311.09083v1
80. Pease, M.C., Shostak, R.E., Lamport, L.: Reaching agreement in the presence of faults. Journal of the ACM **27**(2), 228–234 (1980). https://doi.org/10.1145/322186.322188
81. Protocol Labs: Expected consensus. Filecoin Specification (2017), https://spec.filecoin.io/algorithms/expected_consensus/, living specification consulted 15 August 2026
82. Protocol Labs: Fast finality (F3). Filecoin Project source repository (2025), https://github.com/filecoin-project/go-f3/tree/v0.8.9, version 0.8.9, 30 July 2025; README records mainnet activation on 29 April 2025
83. Putnik, M., Decouchant, J.: Herring: Parallel batch-order-fairness on DAG-based blockchain consensus (2026), https://arxiv.org/abs/2605.23648v1, version 1, 22 May 2026
84. Raikwar, M., Polyanskii, N., Müller, S.: SoK: DAG-based consensus protocols. In: 2024 IEEE International Conference on Blockchain and Cryptocurrency. pp. 1–18 (2024). https://doi.org/10.1109/ICBC59979.2024.10634358
85. Ranchal-Pedrosa, A., Kang, D., Giridharan, N., Malkhi, D., Sadoghi, M., Marsh, B.: Hermes: Low tail-latency via Prefix Consensus (2026), https://arxiv.org/abs/2607.25916v2, version 2, 29 July 2026
86. Ranchal-Pedrosa, A., Marsh, B., Kokoris-Kogias, L., Sonnino, A.: Sedna: Sharding transactions in multiple concurrent proposer blockchains (2025), https://arxiv.org/abs/2512.17045v2, version 2, 29 December 2025
87. Resnick, M.: BRAID: Implementing multiple concurrent proposers. Devcon 7 lightning talk (2024), https://archive.devcon.org/devcon-7/braid-implementing-multiple-concurrent-proposers/
88. Resnick, M.: Breakpoint 2025: Anza block: Max Resnick. Breakpoint 2025 presentation, published by Solana (2025), https://www.youtube.com/watch?v=7kPJ0C9gYSg, presentation of Supernova Part One
89. Rondelet, A., Kilbourn, Q.: Mempool privacy: An economic perspective (2023), https://arxiv.org/abs/2307.10878v3, version 3, 24 October 2023

90. Roughgarden, T.: Transaction fee mechanism design. In: Proceedings of the 22nd ACM Conference on Economics and Computation. p. 792 (2021). https://doi.org/10.1145/3465456.3467591
91. Saraf, A., Kaklamanis, I., Wadhwa, S., Elsheimy, F.: Price of censorship: Censorship resistance and throughput under rational concurrent proposers (2026), https://arxiv.org/abs/2607.16995v3, version 3, 10 August 2026
92. Scaffino, G., Resnick, M., Neu, J.: Gatling: Rapid-fire consensus from parallel composition (2026), https://arxiv.org/abs/2606.18220v1, version 1, 16 June 2026
93. Shoup, V.: Blue fish, red fish, live fish, dead fish. Cryptology ePrint Archive, Paper 2024/1235 (2024), https://eprint.iacr.org/2024/1235
94. Shrestha, N., Shrothrium, R., Kate, A., Nayak, K.: Sailfish: Towards improving the latency of DAG-based BFT. In: 2025 IEEE Symposium on Security and Privacy. pp. 1928–1946. IEEE (2025). https://doi.org/10.1109/SP61157.2025.00021
95. Shutter Network: The first beta release of the Shutterized Gnosis Chain is now live! Project deployment announcement (2024), https://blog.shutter.network/shutterized-gnosis-chain-is-now-live/, deployment announcement, 11 July 2024, for the optional encrypted-transaction path; it does not imply that all Gnosis transactions use that path
96. Shutter Network: Shutter roadmap 2024 (and beyond). Project roadmap (2024), https://blog.shutter.network/shutter-roadmap-2024-and-beyond-2/
97. Sompolinsky, Y., Lewenberg, Y., Zohar, A.: SPECTRE: A fast and scalable cryptocurrency protocol. Cryptology ePrint Archive, Paper 2016/1159 (2016), https://eprint.iacr.org/2016/1159
98. Sompolinsky, Y., Wyborski, S., Zohar, A.: PHANTOM and GHOSTDAG: A scalable generalization of Nakamoto consensus. Cryptology ePrint Archive, Paper 2018/104 (2018), https://eprint.iacr.org/2018/104
99. Spiegelman, A., Arun, B., Gelashvili, R., Li, Z.: Shoal: Improving DAG-BFT latency and robustness. In: Clark, J., Shi, E. (eds.) Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 14744, pp. 92–109. Springer (2025). https://doi.org/10.1007/978-3-031-78676-1_6, proceedings of FC 2024
100. Spiegelman, A., Giridharan, N., Sonnino, A., Kokoris-Kogias, L.: Bullshark: DAG BFT protocols made practical. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 2705–2718 (2022). https://doi.org/10.1145/3548606.3559361, https://arxiv.org/abs/2201.05677
101. Spiesberger, P., Hartenstein, H.: Accountable transaction inclusion lists: Enhancing Ethereum's censorship resistance (2026). https://doi.org/10.48550/arXiv.2607.22040
102. Stathakopoulou, C., David, T., Pavlovic, M., Vukolić, M.: Mir-BFT: Scalable and robust BFT for decentralized networks. Journal of Systems Research **2**(1) (2022). https://doi.org/10.5070/SR32159278, https://escholarship.org/uc/item/36g369xq, peer-reviewed solution article; the author preprint first appeared in 2019
103. Stathakopoulou, C., Pavlovic, M., Vukolić, M.: State-machine replication scalability made simple. In: Proceedings of the Seventeenth European Conference on Computer Systems. pp. 17–33 (2022). https://doi.org/10.1145/3492321.3519579
104. Stouka, A.P., Ma, J., Thiery, T.: Multiple proposer transaction fee mechanism design: Robust incentives against censorship and bribery (2025), https://arxiv.org/abs/2505.13751v3, version 3, 31 May 2026

105. Thiery, T., D'Amato, F., Ma, J., Monnot, B., Tsao, T., Kaufmann, J., Song, J.: EIP-7805: Fork-choice enforced inclusion lists (FOCIL). Ethereum Improvement Proposal (2024), https://eips.ethereum.org/EIPS/eip-7805, draft specification consulted 15 August 2026
106. Thiery, T., Monnot, B., D'Amato, F., Ma, J.: Fork-choice enforced inclusion lists (FOCIL): A simple committee-based inclusion list proposal. Ethereum Research forum post (2024), https://ethresear.ch/t/fork-choice-enforced-inclusion-lists-focil-a-simple-committee-based-inclusion-list-proposal/19870, posted 19 June 2024
107. Tonkikh, A., Arun, B., Xiang, Z., Li, Z., Spiegelman, A.: Raptr: Prefix Consensus for robust high-performance BFT (2025), https://arxiv.org/abs/2504.18649v2, version 2, 29 April 2025
108. Vafadar, M.A., Khabbazian, M.: Condorcet attack against fair transaction ordering. In: 5th Conference on Advances in Financial Technologies. Leibniz International Proceedings in Informatics, vol. 282, pp. 15:1–15:21 (2023). https://doi.org/10.4230/LIPIcs.AFT.2023.15
109. Wadhwa, S., Ma, J., Thiery, T., Monnot, B., Zanolini, L., Zhang, F., Nayak, K.: AUCIL: An inclusion list design for rational parties. Cryptology ePrint Archive, Paper 2025/194 (2025), https://eprint.iacr.org/2025/194
110. Wadhwa, S., Yaish, A., Zhang, F., Nayak, K.: Perils of parallelism: Transaction fee mechanisms under execution uncertainty. In: 35th USENIX Security Symposium (USENIX Security 26). pp. 1087–1105. USENIX Association, Baltimore, MD (Aug 2026), https://www.usenix.org/conference/usenixsecurity26/presentation/wadhwa
111. Wahrstätter, A., Ernstberger, J., Yaish, A., Zhou, L., Qin, K., Tsuchiya, T., Steinhorst, S., Svetinovic, D., Christin, N., Barczentewicz, M., Gervais, A.: Blockchain censorship. In: Proceedings of the ACM Web Conference 2024. pp. 1632–1643 (2024). https://doi.org/10.1145/3589334.3645431
112. Xiang, Z., Tonkikh, A., Spiegelman, A.: Prefix Consensus for censorship resistant BFT (2026), https://arxiv.org/abs/2602.02892v2, version 2, 16 July 2026
113. Xue, B., Laferriere, S., Deb, S., Kannan, S.: BigDipper: Sharded censorship resistant data availability for leader-based BFT (2026), https://arxiv.org/abs/2307.10185v4, version 4, revised 2 June 2026; preprint
114. Yang, L., Park, S.J., Alizadeh, M., Kannan, S., Tse, D.: DispersedLedger: High-throughput byzantine consensus on variable bandwidth networks. In: 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). pp. 493–512. USENIX Association, Renton, WA (Apr 2022), https://www.usenix.org/conference/nsdi22/presentation/yang
115. Yin, M., Malkhi, D., Reiter, M.K., Gueta, G.G., Abraham, I.: HotStuff: BFT consensus with linearity and responsiveness. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. pp. 347–356 (2019). https://doi.org/10.1145/3293611.3331591, https://malkhi.com/files/HotStuff-PODC2019.pdf
116. Yu, H., Nikolic, I., Hou, R., Saxena, P.: OHIE: Blockchain scaling made simple. In: 2020 IEEE Symposium on Security and Privacy. pp. 90–105 (2020). https://doi.org/10.1109/SP40000.2020.00008
117. Zhang, H., Merino, L.H., Qu, Z., Bastankhah, M., Estrada-Galiñanes, V., Ford, B.: F3B: A low-overhead blockchain architecture with per-transaction front-running protection. In: 5th Conference on Advances in Financial Technologies. Leibniz International Proceedings in Informatics, vol. 282, pp. 3:1–3:23 (2023). https://doi.org/10.4230/LIPIcs.AFT.2023.3

118. Zhang, J., Yang, S., Sonnino, A., Loza, S., Kate, A.: Lifefin: Escaping mempool explosions in DAG-based BFT (2025). https://doi.org/10.48550/arXiv.2511.15936

A Review Protocol and Evidence Coding

Discovery process. We began with known work on MCP, Byzantine agreement, DAG consensus, inclusion list, hidden order flow, and parallel replication literatures. We followed their references and traced predecessor and successor protocols to identify additional work. We also inspected standards proposals and official specifications when an implemented or proposed configuration differed materially from its accompanying research paper. This process was iterative. A newly encountered reconciliation rule was used to search its protocol lineage until further additions either duplicated an existing configuration or fell outside the scope in Section 2. We did not use a preregistered Boolean query or claim exhaustive coverage of every database.

Eligibility and configuration identity. A direct MCP configuration is eligible when it satisfies Definition 1. Selected proposal bodies or binding references must remain mandatory constituents through the membership decision. It is not enough that several parties provide transactions, votes, or validity evidence to one assembler. An adjacent constrained assembly, component, or boundary case is eligible only when it isolates a property that would otherwise be conflated with MCP or supplies evidence needed for a comparison endpoint. Ordinary sharding across different state transitions, parallel execution without independent proposal rights, and unrelated MEV mechanisms are excluded. Among eligible designs, we select configurations that instantiate a distinct reconciliation, failure, or recovery rule, establish a comparison endpoint, or mark a boundary in the closest prior literature. Different versions of one named protocol are merged unless a change affects the comparison. Different compositions are separated when an overlay changes hiding, selection, recovery, ordering, or finality. The 39 rows are therefore configurations rather than claims of distinct protocol ancestry.

Lineage checks and exclusions. Several visible lineages were examined without receiving separate rows in the evidence matrix. Mencius, FnF-BFT, BigBFT, and ALDER are important predecessors for replicated logs with multiple leaders, partitioned request assignment, and multiplexed consensus [66,6,3,57]. Dumbo-NG decouples continuing broadcast by each sender from asynchronous agreement and adds a service result for an honest broadcaster [39]. We retain Dumbo2 and DispersedLedger as representative matrix points for common subset selection and continuing author histories, so Dumbo-NG is cited as a material successor in the persistent stream family rather than treated as equivalent to either one. Shoal, BBCA-Chain, Sailfish, and Mahi-Mahi extend the causal DAG lineage through different leader, broadcast, certification, and asynchrony choices [99,64,94,48]. Lifefin is a liveness and recovery overlay with bounded resource use for DAG protocols, not a new outermost reconciliation operator [118]. FairFIL is accountable

constrained assembly machinery whose omission analysis is economic under its stated behavior model [101]. These choices provide representative configurations rather than exhaustive coverage, while identifying relevant work outside the comparison tables.

Source snapshots. The source snapshot was frozen on 9 September 2026. For an archival paper or stable preprint, the coded source is the version named by the bibliography entry. For a mutable specification, improvement proposal, or implementation document, we retain the revision or date exposed by the source when it is material to a claim. A later revision does not silently alter an existing row. When no stable revision can be identified, the relevant claim is treated as provisional and no stronger guarantee is inferred from it. Implementation benchmarks are tied to the reported commit or release when the source provides one. If it does not, that limitation remains part of the measurement note.

Public evidence record. We represent each evidence matrix judgment by the following tuple.

$$(\mathsf{configuration}, \mathsf{source\ snapshot}, \mathsf{scope}, \mathsf{class}, \mathsf{model}, \mathsf{claim}, \mathsf{premises}, \mathsf{endpoint}, \mathsf{source\ evidence}, \mathsf{qualification}).$$

The configuration field identifies the precise protocol composition. The source snapshot fixes the paper, specification, or release. Scope distinguishes direct MCP from adjacent constructions, components, and boundary comparisons. Class records the reconciliation operator; for a component, it identifies the stated composition in which that component is used. It does not import that composition's guarantees. Model records the fault threshold, adversary, network assumptions, cryptographic assumptions, and any capacity premise. Claim and endpoint identify exactly what is established and when. Where a source provides stable numbering, a note identifies the controlling theorem, protocol rule, algorithm, experiment, or explicit negative analysis. Otherwise the row citation and qualification identify only the controlling source and scope of the claim. Qualification records inherited guarantees, unresolved composition obligations, conflicting results, and differences between the source metric and our normalized endpoint. The row and its adjoining note instantiate this public record. We do not imply the existence of a separate page level extraction data set.

Evidence hierarchy. Formal safety, progress, inclusion, hiding, and last-look guarantees require a theorem or a sufficiently complete protocol argument. We also record direct deductions from specified protocol rules, with the premises and argument in the row note. For a negative last-look assessment, the argument identifies a permitted reaction after another input is fixed. Absence of encryption alone does not establish such a reaction. A theorem proved for static corruption is not coded as adaptive security. An availability threshold is likewise not coded as durable retrievability unless the source states the premises about possession, retention, serving, and the timing of corruption needed for recovery. An official

specification can establish a validation or selection rule, but does not receive a fault tolerance theorem that it does not itself provide or inherit from a named base protocol. Performance claims retain the source's workload, hardware, fault setting, unit, and start and end events. Analytical bounds are not presented as measurements. Ordered transactions or transactions ready for execution are not relabeled as executed transactions. An external counterexample or specification audit can qualify a positive claim. In that case both sources and the disputed scope are stated, as in the Ladon and Tusk notes.

Composition and missing evidence. A guarantee is native when the evaluated configuration establishes it directly. It is inherited when a named base protocol supplies it under compatible assumptions. It is composed only when the required interfaces and premises across layers are identified. Otherwise it is not established. Data availability does not imply ordering safety, canonical inclusion does not imply successful execution, and threshold reconstruction does not by itself imply timely plaintext availability. *Not reported* identifies missing quantitative evidence; *unspecified* identifies a missing rule or integration. These labels do not assert that no implementation or later version can provide the property. A stated conditional guarantee is reported with its actual admission premise, including admission to a published block.

A.1 Property and measurement conventions

Safety and liveness. Safety prevents correct validators from finalizing incompatible outputs. The tables state the tolerated Byzantine bound, such as $3f < n$ or $5f < n$, rather than a generic affirmative label. Here n counts validators and f bounds the Byzantine count. Stake or hash power fractions are marked separately. The usual authentication and cryptographic assumptions remain in force. For most BFT rows, safety does not require timely delivery. Probabilistic chain guarantees additionally depend on the stated network and confirmation model. Liveness requires its own network condition. *PSync* means progress after an unknown global stabilization time (partial synchrony), *Async* means asynchronous, randomized eventual progress, and *Sync* means a network with a known delay bound is required. A deterministic delay bound is stronger than eventual termination but requires more timing information. An inherited guarantee is identified by its base protocol. An unresolved construction is marked *Unspecified*. The notes also state how failures couple. They report whether one missing proposal delays a whole step or only its own path. Cadence's slot fallback and Autobahn's advancing lane cuts illustrate why continued consensus alone does not describe performance under faults [8,42].

Goodput and communication. Executed goodput counts distinct useful transactions or state transitions per unit time, excluding duplicate carriage and failed conflicts. The workload must define useful work. Communication claims retain the source's own unit and state whether the quantity is aggregate, per participant, an object size, a message count, or a factor per byte. They also name the

endpoint reached by the payload. Traffic needed for dissemination, agreement, reveal, and recovery is not silently omitted when the source reports it. Malicious ingress is reported separately. A bound linear in proposed payload can still be expensive when duplication, conflicts, or failed execution make useful delivered payload much smaller. The reviewed sources do not expose sufficiently uniform traffic and measurements of useful payload to calculate a defensible amplification ratio across protocols, so we report their raw asymptotic or measured quantities instead. A communication result ending at selection or ordering is not described as bandwidth efficiency for executed output.

Latency from submission to endpoint. For a transaction x, latency to endpoint E is

$$L_E(x) = t_E(x) - t_{\text{submit}}(x).$$

Queueing, proposal formation, ordering, recovery, reveal, and execution can all contribute, sometimes in parallel. A delay measured from a ready proposal omits the earlier user wait. We therefore retain separate columns for protocol ordering bounds and measured latency with its actual endpoints. Appendix E.5 compares the available measurements and failure paths, including their means, tails when reported, and treatment of unfinished transactions. For theoretical bounds, δ is the actual message delay after stabilization and Δ the configured delay bound, with $\delta \leq \Delta$. We use these symbols only when the source's model and start event support that interpretation. A DAG round includes its dissemination work. It is not automatically one δ. We do not use shorthand based on an unnamed baseline or a number of added rounds in place of an absolute endpoint.

Cadence. The cadence records the named production or decision unit and its interval when the source supplies one. Each row identifies the logical unit even when no millisecond value is reported. Examples include an epoch, slot, cut, or DAG round. A sequential epoch may wait for the previous one, whereas pipelined slots overlap. Starting a slot every 100 ms is therefore compatible with a 219 ms average finalization delay [8]. Constellation's 50 ms economic cycles group transaction opportunities within a larger consensus slot. That cycle is neither slot finality nor latency from submission to execution [5].

Reading the tables. Short cells state the bound, endpoint, mechanism, or absence of reviewed evidence. The notes retain the scope and material premises. The evidence matrices do not compute an overall score or rank protocols whose adversary models, premises, and endpoints differ. A result in one column supplies no result in another. Italic entries identify evidence status or provenance rather than service strength. A missing measurement is not an impossibility result. Appendix G defines recurring terms.

Limitations. Expert seeding and reference chaining can miss work outside the reached literatures and naming conventions. The purposive corpus cannot support claims about how common a class is. We did not preregister the search or

perform blinded independent duplicate coding, and we report no statistic on agreement between raters. Class and endpoint assignments therefore remain reasoned judgments that readers may challenge against the cited source and operational rules. The tables do not convert those judgments into numerical scores. Frozen source versions, explicit predicates, normalized endpoints, and retained qualifications make the judgments inspectable but do not eliminate interpretation. The same evidence rules apply to every configuration. Design targets and private information do not upgrade a public claim.

B Formal Interface and Failure Semantics

This appendix formalizes the architecture in Figure 1. The notation describes a common interface for comparison; it does not require protocols to use the same message format or divide their implementation into these functions.

B.1 Proposal records and agreed membership

For ledger step s, write an authenticated proposal record as

$$p = (\mathsf{author}, \mathsf{tag}, \rho, \mathsf{meta}, \mathsf{auth}).$$

The author identifies the proposal path; the tag identifies a step or local sequence position; ρ binds the record to its body B_ρ; metadata carries protocol-specific information such as ordering priorities; and authentication binds the record to its author. The body contains transactions and validation data. It may be sent inline, separately, or as coded fragments. When several authors carry identical bytes, the proposal identity remains part of the record wherever authorship affects selection, ordering, or payment.

Let D_s be the agreed decision. Some decisions directly list their selected proposals. Others name a lane head or graph anchor whose interpretation requires earlier proposal records. Let $\mathcal{H}_s^v$ be the authenticated records available to validator v. We write

$$\mathcal{H}_s^\star = \mathsf{BindHistory}_s(D_s, \mathcal{H}_s^v)$$

for the records that the decision and its interpretation rules permit to affect this step. This expression is evaluated once the required records are available. Correct validators must obtain the same relevant history, or any differences must leave selection, ordering, and execution context unchanged. For example, a reference to a lane head can bind its preceding lane blocks; an unrelated proposal observed only by v cannot alter the decision.

Canonical contribution membership is then

$$S_s = \mathsf{Select}_s(D_s, \mathcal{H}_s^\star).$$

The members of S_s are the selected proposal references. This is checkpoint 3 in Section 2.4. It does not assert that every validator already possesses the referenced bodies.

B.2 Locally recovered bodies and execution readiness

To describe recovery, imagine a lookup table at validator v: a selected reference is entered only when v has received or reconstructed its body and verified that the body matches the reference. This is the *partial body map*

$$\mathcal{B}^v_{s,t} \colon S_s \rightharpoonup \mathsf{Bodies}.$$

The partial-function arrow means that some selected references may have no entry yet. The *domain*, written $\mathrm{dom}(\mathcal{B}^v_{s,t})$, is the set of references whose bodies have been recovered and verified by time t. The references still missing their bodies form the *unresolved set*

$$U^v_{s,t} = S_s \setminus \mathrm{dom}(\mathcal{B}^v_{s,t}).$$

For example, if $S_s = \{\rho_A, \rho_B\}$ and v has only B_{ρ_A}, then the map has one entry and $U^v_{s,t} = \{\rho_B\}$. Another validator may already have both bodies. The selected set is common; local recovery progress need not be.

Let τ^v_s be the first time validator v can interpret the step under the protocol's recovery and failure rules. Usually this requires $U^v_{s,\tau^v_s} = \varnothing$. A protocol that permits a failed contribution to have some other outcome must define that outcome consistently for all correct validators. A local timeout alone cannot authorize incompatible decisions to skip or execute the same contribution. Encrypted configurations also need the opening or decryption information required by their ordering and execution rules.

For configurations that produce a transaction sequence, write

$$\begin{aligned} X^v_s &= \mathsf{Linearize}_s\Big(D_s, \mathcal{H}^\star_s, \mathcal{B}^v_{s,\tau^v_s}, U^v_{s,\tau^v_s}\Big), \\ T_s &= \mathsf{Dedup}_s(X_s). \end{aligned}$$

Correct interpretation requires all completed X^v_s to agree on a common sequence X_s. The protocol's duplicate and replay rules then give T_s. Let Σ_s be the common state before step s, let $\mathcal{L}_{<s}$ be its preceding canonical ledger history, and let I_s collect the agreed decision, relevant authenticated records, and verified bodies or agreed failure outcomes needed to interpret the step. The canonical execution context and state transition are

$$\begin{aligned} C_s &= \mathsf{Context}_s(\mathcal{L}_{<s}, \Sigma_s, I_s, T_s), \\ (\Sigma_{s+1}, R_s) &= \mathsf{Execute}(\Sigma_s, T_s, C_s). \end{aligned}$$

The context supplies every environmental input that can affect execution, including applicable timestamps, block numbers, fee parameters, gas limits, fee recipients, and randomness. The protocol must determine which agreed metadata supply these values when several proposals form one ledger step. If native semantics assign different contexts to different contributions, C_s specifies the context for each retained transaction. These assignments follow canonical contribution identities and the duplicate and filtering rules. The input record I_s therefore retains the provenance needed to identify the surviving occurrence.

Context derivation is deterministic from these canonical inputs. Its required data can become available after ordering finality. Validators execute only after the preceding state and the required context inputs are available. Equal transaction sequences and prior states imply equal execution results only when the execution contexts and transition semantics also agree. The explicit block context in Giga's state transition is one instance of this requirement [68]. The outcomes R_s include failure or reversion where the application defines them. Different validators can reach readiness at different times without disagreeing on the result.

Finality of D_s does not give a finite τ_s^v unless the recovery and reveal assumptions establish one. A protocol may also fix reference order before recovering the bodies. We report that earlier endpoint separately from transaction delivery and execution. Boundary systems supplying only pairwise precedence are not credited with a general total-order interface.

B.3 Transaction identity, duplicate copies, and replays

Let $\mathsf{id}_{\mathrm{tx}}(x)$ be the identifier assigned to transaction x by the protocol. It is commonly a collision-resistant digest of the canonical signed encoding and chain or domain context. Let Seen_s contain the identities excluded by the protocol's replay rule on earlier history. For a rule that removes replays and retains the first occurrence within $X_s = (x_1, \ldots, x_m)$,

$$\mathsf{Dedup}_s(X_s) = \big(x_i \mid \mathsf{id}_{\mathrm{tx}}(x_i) \notin \mathsf{Seen}_s \cup \{\mathsf{id}_{\mathrm{tx}}(x_j) \mid j < i\}\big).$$

The first set removes already processed identities; the second removes repeated copies within the current sequence. Other deterministic duplicate or replacement rules must be stated explicitly. Where a source records redundant entries and rejects them during execution, its actual endpoint and accounting are preserved rather than silently rewriting the ledger.

Different signed transactions expressing the same economic intent remain distinct unless the protocol defines an intent identity and replacement rule. Likewise, two transactions using the same balance or nonce can be distinct conflicts. They are resolved by the protocol's validation and execution semantics, not by assuming their identifiers coincide.

For configurations using the filtered sequence above, a transaction is canonically included when a copy survives into T_s. Removing other copies does not undo its inclusion. This endpoint says nothing about whether execution succeeds. Fees and rewards may attach to earlier work, including duplicate carriage, as discussed in Section 7.

B.4 Formal inclusion claims

A bounded inclusion claim names a protected transaction or proposal z, a receipt and admissibility premise $\mathcal{R}(z)$, an endpoint E, a horizon w, and a failure probability ε:

$$\Pr[E_{\le w}(z) \mid \mathcal{R}(z)] \ge 1 - \varepsilon.$$

The horizon is measured from the claim's stated start event and in its stated unit, such as elapsed time, slots, or epochs. A bound in asynchronous epochs need not be a bound in wall-clock time.

An include-or-empty claim instead uses

$$\Pr[E_s(z) \vee \mathsf{Empty}_s \mid \mathcal{R}_s(z)] \geq 1 - \varepsilon_{\mathrm{sel}},$$

where Empty_s means the relevant canonical transaction output for step s is empty. The rule prohibits a specified selective omission but permits an arbitrarily long run of empty outputs unless progress is bounded separately.

To make that extra condition precise, let $\mathcal{R}^{\mathrm{persist}}_{s,b}(z)$ state that the required receipt, admissibility, and resubmission premises persist for b steps starting at s. Suppose that, under this same conditioning, each of those steps violates its include-or-empty rule with probability at most $\varepsilon_{\mathrm{sel}}$, and

$$\Pr\left[\bigwedge_{j=0}^{b-1} \mathsf{Empty}_{s+j} \mid \mathcal{R}^{\mathrm{persist}}_{s,b}(z)\right] \leq \varepsilon_{\mathrm{run}}.$$

A union bound then gives

$$\Pr[E_{\leq b}(z) \mid \mathcal{R}^{\mathrm{persist}}_{s,b}(z)] \geq 1 - \varepsilon_{\mathrm{run}} - b\varepsilon_{\mathrm{sel}}.$$

Here $E_{\leq b}$ means that E is reached within that b-step window. No independence between steps is needed. The per-step bounds must apply under the persistent premise used above; a bound under a different conditioning event cannot simply be substituted.

The table labels preserve these distinctions. *Next slot, height, or cycle* identifies a bounded endpoint under the accompanying note's premise. *Include-or-empty* gives no separate bound on empty runs. *Aggregate only* protects a qualifying set without individual service. *Eventual* gives no fixed horizon, and *Economic* denotes deterrence under an incentive model. A support threshold is stated directly where available: Multiplicity's rule makes a transaction present in more than one third of committee bundles mandatory in every valid assembly containing two thirds of the bundles [38].

B.5 Formal hiding claims

For an observer coalition $\mathcal{A}$ and a cutoff τ, semantic content hiding requires two admissible bodies B_0 and B_1 with the same permitted leakage to produce computationally indistinguishable views before τ:

$$\mathsf{View}^{<\tau}_{\mathcal{A}}(B_0) \approx_c \mathsf{View}^{<\tau}_{\mathcal{A}}(B_1).$$

The observers, cutoff, security assumptions, and leakage are part of the claim. GNR excludes the initial receiving proposer from its observer guarantee. Cadence protects a designated proposer from reacting through the construction of new

proposals [41,8]. Sedna's bound on exposed bits is recorded separately from this indistinguishability notion [86].

The protected action also matters. Selecting an encrypted batch, fixing its order, constructing a fresh proposal, and choosing whether to open a prior commitment have different binding points. HoneyBadgerBFT selects encrypted batches before opening, while Fino orders before opening [71,65]. A guarantee for one action does not automatically cover later candidate selection or reactions to public information.

B.6 Failure and recovery by reconciliation class

This section records how each reconciliation class treats missing proposals and unavailable selected bodies. These policies are separate from the operator that fixes canonical contribution membership.

Constrained assembly

Missing proposals. There is no reserved vector entry for each constraint contributor. Missing messages instead affect which constraints apply. A transaction in too few eligible FOCIL lists, invalid in the resulting state, or beyond available capacity need not be included. Conversely, requiring every locally observed transaction could prevent progress under inconsistent views or excess demand. Liveness depends on propagation, aggregation, capacity, and the base consensus protocol. Safety depends on the base protocol and the correct application of its validity or support rules.

Recovery of selected bodies and tradeoffs. Body delivery normally follows the base block protocol. The leader still constructs the assembly, sees public lists, and generally controls placement. These designs can therefore constrain omission with a modest architectural change, but do not by themselves provide hiding, fair ordering, or relief from the leader's dissemination bottleneck. The fee analysis of Stouka, Ma, and Thiery assumes transactions of equal size, user values independent of position, myopic Bayesian participants, and specified bribe functions and beliefs. Its incentive guarantees restrict admissible bribes and other participants' strategies. User guarantees also restrict overbidding and demand above the base fee plus processing cost. General agreements made outside the protocol are not covered by that model [104]. Section 7 explains why these qualifications matter.

Fixed proposal vector

Missing proposals. A configuration must say whether an unfilled position makes the step wait, becomes `nil`, or triggers fallback. Waiting exposes the whole step to a slow proposer. Fallback can add communication and latency. A rule based on a ranked prefix preserves supported entries up to its closing point and excludes the remainder from this step. Cadence illustrates the resulting failure coupling.

A split that blocks a quorum for just one proposer can trigger fallback, while a fully silent proposer need not. Later slots may decide, but delivery of the ledger prefix and stateful execution wait for the unresolved earlier slot [8]. If each of k proposers independently causes such a split with probability q, the chance that at least one does is $1 - (1 - q)^k$. For $k = 5$ and $q = 0.1$, this is about 41%, versus 10% for one proposer. This is an illustrative model, not a measured failure rate, and independence is an explicit assumption. Delaying submission can improve access to recent transactions while increasing deadline risk. Sending earlier gives more dissemination slack but fixes an older transaction set. Cadence discusses this timing incentive and an accountability mechanism requiring the actual Byzantine count to be at most half the configured fault budget. Network delay can also affect correct proposers, so one fallback is not evidence of misbehavior [8]. Appendix E.5 separates the proven fallback bound from the simulation under favorable conditions. BigDipper-7 distinguishes three outcomes: zero marks an excluded validator position, `NULL` marks an honest empty mini block, and `Incomplete` marks a failed DA-CR instance followed by timeout and view change [113].

Recovery of selected bodies and tradeoffs. Every selected body must be recovered under the configuration's availability rule. Deterministic ordering still needs rules for duplicates, nonce conflicts, priority, and payment. Protection against a last look additionally requires that one proposer cannot use another's information while its own relevant choice remains open. Cadence uses threshold encryption to prove that the designated proposer cannot react to other proposals. Encryption for BRAID is a companion mechanism, not a consequence of vector agreement [8,17,96]. For BigDipper-7, a completed Card-7 certificate binds a uniquely recoverable coded object. The native DA-CR retrieval interface supplies eventual full or partial reads through a separate path, while ordering within the object, application ordering, execution, and serving remain duties of upper layers [113]. The architectural benefit is explicit attribution. The corresponding obligation is to resolve each reserved position and recover every selected body.

Selected proposal collection

Missing proposals. Candidates below the selection threshold are omitted, allowing progress without the slowest proposer. This separates aggregate progress from service for a particular transaction. The bound $|S_s| \geq n - f$ does not imply that a specified correct proposer belongs to S_s. GNR's analyzed service after GST permits a globally empty slot rather than selective omission of a timely transaction. AMP instead guarantees proposal selection at the next height under the stronger premise that every correct validator attested it. Constellation likewise requires qualifying proposals in the selected aggregate, subject to receipt thresholds and sufficient aggregate support. Scheduling can defer transactions beyond execution capacity [5]. This is the same kind of service enforced by a threshold, but not automatically AMP's precise theorem for the next height.

Each comparison must retain its attestation and capacity premises. Multiplicity likewise guarantees service only for a transaction carried by enough selected bundles. Raptr supplies no universal horizon for a client transaction merely because its batch was disseminated.

Recovery of selected bodies and tradeoffs. GNR reconstructs bodies from relay fragments. AMP's selected identifiers must resolve to their disseminated bodies. Multiplicity carries the selected bundles in the assembled block, while Raptr's supported prefix is meaningful only for batch identifiers whose bodies meet its availability rule. A possession certificate means only what its signing predicate and adversary model establish. Under a static model with at most f faults, $f + 1$ distinct possession attestations imply that at least one signer is correct. They do not imply that every validator has the body or that the correct signer remains obliged and able to serve it. A model with adaptive or mobile corruption additionally needs an argument that possession, retention, and serving survive the allowed corruption timing and state erasure. Timely recovery also needs a retrieval protocol and network assumptions. A common collection likewise needs separate deduplication, ordering, and execution rules. This family amortizes agreement over a compact collection without waiting for everyone. Collection validity alone establishes neither a particular transaction's inclusion nor its execution time.

Persistent proposal streams

Missing proposals. A missing new lane block need not leave an empty global position. The next cut can repeat that lane's previously selected head while other heads advance. For interleaved ordering instances, the equivalent protection requires a merge rule that can bypass a stalled instance or reassign its request bucket. A merge that waits for the next item from every instance would still block globally. Likewise, a client assigned only to a faulty leader must wait for reassignment or use another eligible route [42,102,103]. DispersedLedger similarly allows an epoch to agree without every current author input. Links between nodes recompute recoverable prefix endpoints for each author in later epochs. A delivered map suppresses repeats, and every correct proposer's block is eventually delivered rather than being silently discarded from its author history [114].

Recovery of selected bodies and tradeoffs. Under Autobahn's stated static fault model, its availability certificate with $f + 1$ votes establishes that at least one signer was correct and held the body when voting. It does not establish possession by every validator, permanent custody, or security under an unstated model with adaptive or mobile corruption. Ordering a compact head can therefore precede body retrieval, leaving retention and recovery in the end-to-end progress argument. Deterministic interleaving fixes order only after prefixes and priority metadata are fixed. It supplies neither hiding nor fairness in receive order. The family's primary gains are sustained parallel production and the opportunity to isolate slow paths. One cut can cover many accumulated lane blocks after a

stall, so agreement need not replay one separate decision per missing production interval. Validators must still process that backlog. Consensus recovery and catching up in execution have different costs [42]. DispersedLedger makes this separation explicit. Agreement may finish over commitments. Linking between nodes then makes additional prior blocks canonical, and AVID-M reconstructs the selected and linked bodies before delivery [114].

Causal proposal DAG

Missing proposals. A late or insufficiently referenced vertex, or a vertex outside the closure, can be absent from one decision yet enter a later anchor's history. Whether this occurs, and with what bound, depends on the reference and commit rules. The DAG shape itself does not guarantee inclusion or liveness. Published analysis identifies liveness issues in particular Tusk and GradedDAG specifications [93]. Lifefin further shows that unchecked mempool growth can prevent commitment even when the underlying DAG protocol continues to exchange vertices. Its fallback using asynchronous common subset (ACS) and recovery with bounded resources are additional liveness premises, not consequences of DAG structure [118].

Recovery of selected bodies and tradeoffs. Certificates and references support recovery according to their dissemination and retention rules. Once bodies are recovered, deterministic traversal gives a common order but may favor an index, fee, timestamp, or reference pattern. Mysticeti's traversal within a round illustrates this separation between agreement and fairness [72]. A shared history can amortize dissemination, voting, and ordering evidence, at the cost of coupling one proposal's selection and priority to references made by others.

B.7 Membership and probabilistic finality

The five comparison classes describe reconciliation structure. Four are the architectural families of direct MCP. The central constrained assembly configurations instead have MCP-adjacent scope. Fixed committees, open membership, stake or work weighting, and deterministic or probabilistic confirmation describe the model in which that structure runs. The final comparison panel groups probabilistic configurations for their shared confirmation questions, while retaining class tags where the boundary system itself defines the corresponding ordering rule. BlockDAG and multichain protocols admit concurrent producers through probabilistic election. Their resistance to Sybil attacks and confirmation assumptions therefore differ from a single committee BFT decision. Their outputs also differ. SPECTRE provides pairwise precedence rather than the total order needed for a general state machine. PHANTOM/GHOSTDAG order a blockDAG. Conflux uses a pivot chain. Prism separates proposer, voter, and transaction blocks. OHIE interleaves Nakamoto chains [97,98,60,9,116]. Filecoin Expected Consensus can elect several producers weighted by storage power into a tipset,

while F3 supplies a separate finality overlay [81,82]. We retain these as boundary comparisons. Concurrent block production alone implies neither deterministic inclusion in the same step nor a common confirmation or execution endpoint. BRAID and Filecoin EC both combine concurrent proposals into a step. The compared BRAID construction reserves tracks within the same step. An EC tipset collects the compatible blocks produced by that epoch's elected miners, with variable membership and possible null epochs [87,81]. The distinction is fixed positions versus an elected set, not simply BFT versus proof of stake. Braided proof of work chains such as Chainweb are a separate design. Neighboring chains reference one another's headers and need their own confirmation across chains [70]. We do not conflate that use of *braid* with Resnick's BRAID protocol. The scope labels identify configurations as direct MCP, MCP-adjacent, component, or boundary comparisons. The reconciliation classes explain how configurations that fix canonical contribution membership combine their inputs. The evidence matrices retain those distinctions for the boundary configurations.

C Protocol Placement by Reconciliation Class

This section records the representative placements and historical relationships used to apply the reconciliation taxonomy.

C.1 Constrained assembly

FOCIL narrows the assembler's authority by making some otherwise valid blocks unsupported. Its committee members gossip inclusion lists in slot N. Attesters support a block in slot $N + 1$ only if it satisfies the lists that contain no equivocation and were stored before the freeze deadline, subject to capacity and validity conditions applied after the block [106,105]. AUCIL studies incentives for constructing and aggregating these lists [109]. FairFIL adds accountable list construction and analyzes the cost of bribing participants to omit a transaction from all lists. That economic deterrence is distinct from a protocol rule that guarantees a timely client transaction enters an eligible list [101]. Unlike Multiplicity, the inclusion lists are not selected as components that carry transactions in the canonical block. They induce a validity condition on a separately assembled block, which is the criterion separating constrained assembly from a direct selected collection.

C.2 Fixed proposal vector

Interactive consistency expresses the classical ideal in which correct parties agree on a vector retaining each correct party's value at its index [80]. Asynchronous variants cannot wait indefinitely to distinguish a slow party from a silent faulty one, so they preserve a common core while permitting missing entries [33,26,14]. Cadence's Chorus decides a meta block with one entry per designated proposer. Its Conductor pipelines successive instances and throttles new starts to bound

work left open during asynchrony [8]. Strong Prefix Consensus agrees on a prefix of a proposal vector ranked separately at each step. Its entries do not extend continuing lanes [112]. Parallel agreement paths for each step can produce the same shape. The Ethereum design with two proposers finalizes a pair of execution payloads and analyzes duplication and tip allocation [78]. BRAID combines k tracks with deterministic ordering across tracks [87]. BigDipper-7 denotes the Navigator, Card-7, Scooping, and HotStuff-2 composition. It reserves one systematic position per validator mini block and fills an excluded position with zero before a leader encodes the vector again for sharded availability. Because authenticated mini blocks remain separately represented in the certified object rather than merely constraining another payload, we classify its integrated HotStuff-2 configuration as fixed vector [113]. In the BRAID configuration the k tracks jointly form the output of one step. They do not independently advance reusable lane heads as in Section 3.3. The Multiple Byzantine Agreement (MBA) design also runs several agreement paths for the same step [88]. Its placement follows this output in vector form. A concrete instantiation must still specify how each path closes when its proposer is missing.

C.3 Selected proposal collection

The constructions of Ben-Or, Canetti, and Goldreich, and of Ben-Or, Kelmer, and Rabin, combine reliable broadcast from each proposer with agreement on a common core [14,15]. This asynchronous common subset (ACS) pattern underlies HoneyBadgerBFT, which adds coded broadcast and threshold encryption. Dumbo, BEAT, MiB, and FIN explore different broadcast, agreement, cryptographic, and resilience costs. The evidence matrix uses Dumbo2 [71,20,44,43,34,61,35]. Red Belly's Set Byzantine Consensus decides a valid subset without conflicts, with transaction ordering defined by the application [28]. Agreement on a single value does not itself provide this construction. Cachin et al., the work of Borran and Schiper, and DBFT require a wrapper for multiple proposals [18,16,4,27]. Certificate maps need not expose one agreement instance per proposer. GNR encodes batches into hiding fragments, has sampled relays attest receipt, and decides an attestation map before reconstruction and ordering [41]. AMP broadcasts plaintext bodies and carries proposal identifiers in Tendermint PRECOMMIT vote extensions. The certificate at the next height determines the selected collection [21]. Supernova Part One and Constellation also use receipt thresholds [88,5]. Constellation's attestations cover a rolling window of proposal commitments. Its batch rule selects newly attested proposals, excluding those already accounted for in earlier batches. This remains collection selection: retaining candidates across cycles does not make the selected object a lane prefix or a causal closure. Multiplicity has committee members sign bundles that carry transactions from their local mempools. The leader selects at least two thirds of those bundles for the block, so the bundles themselves are canonical contributions rather than evidence that merely constrains an unrelated assembly [38]. We therefore classify Multiplicity as a direct selected collection. Raptr is the ordered variant. Validators disseminate independently authored transaction batches, a rotating leader selects

and orders their identifiers, and validators support nested available prefixes of that ordered proposal [107]. Leader control over membership and order weakens its service properties, but does not change the reconciliation operator.

C.4 Persistent proposal streams

Systems with independent lanes separate production from ordering. Autobahn certifies lane batches with availability proofs and agrees on a cut of the lane heads [42]. DispersedLedger opens dispersal and agreement instances for each author in each epoch. Each epoch first delivers the fresh collection selected by binary agreement and then uses committed completion observations for each author to recover previously undelivered prefix blocks. Because the full rule for canonical inclusion depends on continuing author histories rather than only the fresh set, we classify it as a persistent stream [114]. Dumbo-NG likewise runs continuing broadcasts from each sender and uses asynchronous agreement to select certified stream indices. Its service guarantee for an honest broadcaster under the quality condition is a material result for persistent streams, although it is not duplicated as a separate matrix row [39]. Sei Giga combines this architecture with pipelined EVM execution and later state attestation. Its fee and duplicate rules belong to that complete system [68]. Hermes is a finality protocol over an abstract prefix space. Its instantiation for a single chain is a finality gadget, while its instantiation for multiple lanes reconciles lanes organized as in Autobahn using a heaviest common prefix. The latter is the configuration for persistent streams scored here. Multimmit votes per proposer chain on its furthest supported head [85,59]. Both rules can retain a supported prefix when a newer full cut lacks support. Their latency claims remain specific to each configuration. A second pattern interleaves persistent ordering instances. Gatling staggers K atomic broadcast instances, reducing the time between proposals without reducing component confirmation latency [92]. RCC runs concurrent instances with a primary and a backup [45]. Mir-BFT and ISS assign request buckets to leaders or segments and rotate assignments [102,103]. Ladon uses monotone ranks for decisions arriving at different rates, while Orthrus separates partially orderable transactions from those requiring global order [63,62]. These components may already finalize local order before interleaving. Their shared question is whether one slow instance can hold back the combined log.

C.5 Causal proposal DAG

Dependencies among DAG vertices can create the *hangover* described by Autobahn. After a network stall, a validator may need several dependent ancestor fetches before it can validate the newest vertex or advance. Independent lanes let cut agreement proceed alongside recovery and cover accumulated history compactly [42]. The DAG's benefit is that references also carry shared causal, voting, and inclusion evidence. The broadcast obligation matters. Consistent broadcast prevents conflicting deliveries at correct validators. Reliable broadcast additionally requires eventual delivery everywhere once any correct validator

delivers [18]. Meeting this totality condition can add dissemination and recovery work. Autobahn's availability certificate with $f + 1$ votes is weaker than either abstraction. It establishes a correct data holder but does not rule out equivocation. Agreement must reconcile conflicting certified lane views. Thus the distinction is not that every DAG is expensive or every lane implements consistent broadcast. Instead, it is where consistency, totality, and recovery are enforced [42,31]. DAG-Rider implements asynchronous atomic broadcast over a DAG organized into rounds [50]. Narwhal supplies the certified graph used by Narwhal–HotStuff, Tusk, and Bullshark. We group these compositions with causal DAGs. The Narwhal–HotStuff row includes HotStuff agreement and the deterministic ordering of committed causal histories [31,100]. Mysticeti embeds votes in vertices and uses a threshold logical clock [7]. Fino's DAG protocol gains blind ordering from a separate layer that orders before opening, not from graph structure alone [65]. Shoal, BBCA-Chain, Sailfish, and Mahi-Mahi continue this lineage with different leader selection, broadcast, certification, and choices for asynchronous DAG construction. These differences matter within the causal DAG family without creating another outermost reconciliation class [99,64,94,48].

D Relation to Prior Work

The main text uses prior protocols to explain the architecture and compare their guarantees. This appendix places the resulting systematization beside broader consensus surveys, MCP theory, and work on individual stages of the transaction path.

Consensus taxonomies and efficiency comparisons. Broad systematizations organize blockchain consensus and its safety, liveness, communication, and performance tradeoffs [11]. Raikwar et al. classify DAG based consensus and examine its security, scalability, and fairness properties [84]. The closest broad comparison to our scope is Alpos et al., who review leader based, parallel leader, DAG based, inclusion list, and consensusless protocols using proposal latency, proposal period, maximum transaction censorship, communication complexity, and reported WAN throughput and latency [2]. Our scope is narrower, but our unit of analysis is finer. We compare concrete configurations by the reconciliation operator that fixes canonical contribution membership. We also separate scope status from reconciliation class and normalize claims to receipt, proposal admission, contribution membership, canonical transaction inclusion, execution, and state finality. This prevents a proposal rate, consensus commit, or component guarantee from being silently read as a complete transaction service guarantee. A useful overlap is BigDipper-7. We code its integrated Card-7/HotStuff-2 configuration as a fixed vector because validator identities are assigned to logical mini block positions before selection and the committed object resolves those positions separately. The classification does not follow merely from the systematic coordinates of its erasure code. We then distinguish canonical contribution membership in the final block from ordering within the object at a higher layer, application ordering, and execution [113].

Selective censorship resistance and hiding. The GNR paper by Garimidi, Neu, and Resnick is the closest protocol level point of departure [41]. It identifies selective censorship resistance and hiding as key properties for on chain auctions and gives a concrete MCP construction that targets both. We treat GNR as an important configuration rather than as a definition of the whole design space. The reconciliation taxonomy places it beside fixed proposal vectors, persistent proposal streams, and causal proposal DAGs, whose decided objects and failure behavior differ. We further analyze bounded inclusion, hiding, deterministic ordering, and last-look protection separately. A design therefore receives credit only for the property and endpoint established by its actual construction.

Censorship, duplication, and throughput. The tension between redundant dissemination and useful throughput already appears in asynchronous common subset and replication designs with multiple leaders [71,102]. Elsheimy et al. formalize a Byzantine transaction assignment model and prove conditional deterministic lower bounds. Optimal throughput incurs f rounds of censorship delay, while constant round inclusion incurs a factor f throughput loss under their stated conditions [36]. They also give intermediate and randomized assignment rules. Saraf et al. study a distinct rational model in which an adversary bribes concurrent proposers. They define economic censorship resistance relative to the user's expected payment and analyze how fee sharing and duplicate inclusion affect both censorship cost and throughput [91]. These works characterize important frontiers within specified assignment and incentive models. We retain those assumptions and locate the result at the layer where it is proved. We then ask whether the assigned data are selected, recoverable, ordered, executed, and paid for in the complete configuration.

Adjacent transaction pipeline work. Other systematizations cover front running defenses, encrypted order flow, transaction ordering, transaction fee mechanisms, and builder markets [12,89,90,25]. These mechanisms can alter one stage without changing the underlying reconciliation structure. We use this literature to audit composition rather than to enlarge the MCP label. The resulting contribution is a reconciliation taxonomy, a normalized set of service endpoints, explicit missing proposal and selected body recovery semantics, and an analysis across layers spanning admission, availability, hiding, ordering, execution, state attestation, payments, and organizational control.

Hiding and fair-ordering mechanisms. Threshold-encryption constructions and the optional Shutter path illustrate delayed reveal, with protection limited to the paths that actually use them [13,117,24,95]. Identity-based and multiparty delayed-encryption designs alter the trust, setup, and timing assumptions; some replace a stable threshold committee with secrecy and timely participation assumptions on other participants [32,74,53]. Their recovery and payment obligations form part of the complete service [89].

Fair-ordering mechanisms protect different notions of order. Wendy uses a clock-cutoff premise. Aequitas, Quick Order Fairness, and Themis use different

receipt-order thresholds and treatments of cycles [58,52,19,51]. FairDAG distributes observation evidence through a DAG. Herring identifies a liveness flaw in the published FairDAG-RL pseudocode and proposes a patch [49,83]. Fairness and progress need to hold for the same concrete ordering rule; the ordering mechanism alone does not supply proposal admission or body recovery.

E Comparison Tables

E.1 Reading the evidence panels

The panels are evidence maps rather than protocol rankings. A bounded inclusion result does not promise successful execution, and a safety threshold says nothing about performance. Measurements with different endpoints remain separate. In the guarantee tables, n is the validator count and f the Byzantine count unless a cell explicitly refers to stake or hash power. Safety and progress assumptions are recorded independently. A safety threshold is not silently transferred to progress. Progress may require a stricter fault bound, partial synchrony, synchronized clocks, static corruption, randomness, data availability, or participation assumptions stated in the cell or note. *PSync*, *Async*, and *Sync* identify the network models in Section 2.6. *Inherited from base protocol* identifies a property supplied by the named consensus protocol. *Plaintext* means that recipients see unencrypted contents; it does not assert that every observer sees them. *No* in a last-look cell refers to the named reactive choice, whose witness or scope is given below. *Unspecified* instead records a missing protocol rule or integration. Content-blind selection or ordering does not close every later action or hide public signals. Stars point to qualifications below each panel.

Guarantee scope. The guarantee panels separate aggregate progress from service for a specified proposal or transaction. A selected collection may satisfy its quorum rule while omitting one correct proposer's input. A constrained assembly may force a qualified transaction into the next block without giving its list authors independent blocks. A component may establish retrievability without deciding ledger order. The cells therefore report only the named property at the named endpoint. They do not convert inherited, conditional, or component claims into complete transaction service.

Failure and recovery scope. The same taxonomy exposes different failure surfaces. A fixed vector must resolve each reserved position by waiting, inserting a default, closing a prefix, or invoking fallback. A selected collection can omit candidates below its selection threshold, but every selected reference still needs a recoverable body. Persistent streams can advance some lanes while retaining older heads on others. A causal DAG can amortize availability evidence across ordering, but a data plane certificate alone does not supply a ledger order. These recovery semantics are part of the configuration, not an implementation detail beneath the class label.

Measurement scope. The performance panels preserve the source's communication unit, start event, end event, workload, and reported rate. They do not place proposal bandwidth, ordered transactions, executed gas, and successful state transitions on one scale. Cadence's fast path bound is stated relative to a slot deadline. Sei Giga's reported ordering latency stops before state attestation. These distinctions explain why a short cadence or a high rate reported by the source cannot by itself establish low user latency or high executed goodput.

E.2 Finding and reading a table entry

Table 4 maps every main-matrix row to two evidence panels. The guarantee panel and its notes explain the protected input, eligibility, fault and network assumptions, inclusion endpoint, and any hiding or last-look claim. The performance panel and its notes explain communication units, payload and committee sizes, start and end events, workloads, and cadence. The protocol-name links in Tables 2–3 lead directly to the relevant guarantee note. This index supplies the same route in print and points to the separate performance evidence.

Table 4. Where to find the full notes for every protocol row. The references identify appendix tables; read the notes following each panel as part of it.

Protocol rows	Guarantees	Performance
Multiplicity, FOCIL, BRAID, MBA, Cadence, GNR, Supernova Part One, Constellation	Table 7	Table 12
AUCIL, BigDipper-7, AMP, Ethereum (two proposers), HoneyBadgerBFT, Dumbo2, Red Belly	Table 8	Table 13
Gatling, Mir-BFT, ISS, Ladon, Orthrus, Raptr, Strong Prefix, DispersedLedger, Autobahn, Sei Giga, Hermes, Multimmit	Table 9	Table 14
Narwhal–HotStuff, Tusk, Bullshark, DAG-Rider, Mysticeti-C, Fino	Table 10	Table 15
SPECTRE, GHOSTDAG, Conflux, Prism, OHIE, Filecoin EC	Table 11	Table 16

Common progress labels. The liveness labels *PSync*, *Async*, and *Sync* name network models, not degrees of liveness. Clock, coin, sampling, data, and view-change assumptions remain in the row notes. Cadence requires synchronized clocks; GNR's relay sampling gives progress with overwhelming probability; Constellation requires its sampled-committee and Votor premises. These are assumptions of the credited guarantees, not reasons to color the same property differently. Narwhal alone disseminates data asynchronously; the compared Narwhal–HotStuff composition needs partial synchrony for ledger progress. Sei Giga inherits Autobahn progress through committed ordering. Ladon's safety

remains disputed, and the analyzed Tusk construction has a refuted liveness claim.

Liveness dependencies and inclusion. The liveness assessment retains the recovery obligations for selected bodies, including any specified deterministic rejection of malformed input. Eventual retrieval does not imply permanent archival storage, a recovery deadline, or immediate full replication. Recovery concerns selected data; *Eventual* inclusion concerns eligible input that has not yet been selected. A stalled ordering layer can therefore retain availability, as the Tusk note explains. A quorum with one correct full-data holder can suffice under static faults and retention; additional holders improve redundancy.

The shared eventual-inclusion label protects a valid transaction admitted by a correct proposer, or a correct proposer's broadcast input where that is the source's interface. Clients must satisfy the stated submission, resubmission, and admission rules. It does not promise inclusion of every available Byzantine proposal. Narwhal uses reinjection when garbage collection omits a block; Autobahn and Sei Giga advance correct lane prefixes; Hermes combines this data service with its specified cut construction; Multimmit proves transaction liveness directly. The Hermes argument below is a composition of published results, not a separately stated theorem in version 2. Multimmit's stronger view-specific inclusion theorem remains in its note. No common label equates these stronger results or extends ordering progress to successful execution.

Selection and service. An *eligible* or *timely* input meets the protocol's named receipt, validity, timing, and capacity premises. *Subset size only* ("aggregate only" in the evidence panel) describes a cardinality guarantee, such as selecting $n - f$ inputs; it does not identify a protected proposer. *Include-or-empty* permits the whole output to be empty, so a bound on consecutive empty outputs is needed before claiming a transaction deadline. *Backlog-bound epochs* in HoneyBadger means a number of epochs depending on the pending queue and batch size; it is not a fixed number of seconds. *Bucket rotation* means changing which leader handles a group of requests, so one faulty owner cannot retain that role forever. Client admission, retransmission, and the source's progress assumptions still apply.

References, recovery, and order. A *supported prefix* is an initial segment backed by the protocol's required votes and data checks. Raptr applies this rule to a leader's list; stream protocols instead select how far to advance continuing lanes. A *cut* names those lane endpoints; merging prefixes turns the chosen contributions into one order. *Linked recovery* uses later references to retrieve earlier omitted blocks. *Causal closure* includes the ancestors required by the DAG's selection rule. *Rank merge* orders contributions using protocol-defined ranks, without by itself establishing receipt-order fairness. *Certified* refers to the precise attestation condition in the row note, not execution or possession by all validators. *Quorum-gated* means that the next round waits for the specified voting or reference threshold; a *wave* groups DAG rounds used by a commit

rule. A missing recovery-time measurement remains NR; it does not negate an established recovery argument.

Information and timing. *Opening* and *reveal* mean making hidden contents available for execution, usually through threshold decryption. Coded fragments alone do not establish hiding: the observer and encoding matter. In GNR, *valency* is the log prefix already forced by the honest state and environment, regardless of future adversarial choices. This can precede validators recognizing the decision. Its hiding claim ends when the transaction's inclusion and log position become inevitable, and excludes the initial receiving proposer [41]. *Blind selection* or *blind ordering* protects the named choice against hidden contents, not against later public information. *Same view* and *next finalized leader* are conditional endpoints, not promises that every view finalizes. *Expected* averages over the stated randomness; *amortized* averages over a sequence of operations. Neither is a per-transaction deadline or a tail percentile.

Reactive choices. The negative last-look entries concern the base configurations in the rows. For proposal, input, payload, vertex, and mining inputs, an admissible execution lets a participant observe an already fixed plaintext input, or a later public signal, before fixing its own fresh input, while both contributions remain eligible for the same unresolved reconciliation step. The input rules do not require every competing contribution to be fixed before that observation; later agreement determines which authenticated contributions become canonical. This is a deduction from the specified input interfaces, not an inference from missing cryptography alone. It applies to the plaintext collection, stream, DAG, and probabilistic rows with these labels. An application may impose an additional commitment rule, but that is not supplied by the compared consensus configuration. Causal precedence can keep a reaction behind its trigger, and a fixed merge can restrict placement; neither makes the initial reaction blind. No profitable attack or fixed reaction interval is inferred from the negative label.

Bundle selection, assembly, leader lists, and fresh cuts identify additional choices over already available inputs. Inclusion obligations constrain these choices without necessarily making them unique. The Multiplicity, FOCIL, BigDipper, Autobahn, and Hermes notes identify the remaining discretion. Giga's row credits its proposed Sedna privacy path; its note separately scopes the plaintext witness and the conditions needed for content protection. The Multimmit entry concerns fresh lane inputs and does not deny its stronger quorum-backed membership guarantee. BRAID and MBA leave the optional hiding and release mechanism to an overlay. Supernova Part One's cited presentation does not specify the complete integration, recovery, hiding, binding, or merge rules needed for a stronger answer.

Table 5: Notation and abbreviations used in Tables 1–3 and the performance tables. Symbols are local to the indicated row; identical letters across papers need not denote the same quantity.

Symbol or label	**Meaning and scope**
k, q in Table 1	k reserved vector positions, candidate proposals, or continuing lanes, according to the column. Where a collection requires q of k candidates, q is a selection threshold, not a validator quorum.
n, f	Validator count and maximum Byzantine count in committee rows. Conditions such as $3f < n$ and $5f < n$ retain each source's other assumptions. In OHIE, n counts miners.
Sync, PSync, Async	Synchronous delivery; partial synchrony after GST; randomized asynchronous progress. GST is the unknown global stabilization time. Data, clock, coin, sampling, and view-change premises remain in the row notes; the label alone does not establish client inclusion.
No; Unspecified; NR; N/A; Base protocol	Named last-look choice remains reactive; protocol rule or integration missing; no quantitative result reported; outside the row's scope; supplied by the named base protocol.
FV, SC, PS, CD, CA	Fixed vector, selected collection, persistent streams, causal DAG, constrained assembly. The scope suffixes adj., comp., and bound. mean adjacent, component, and boundary; bound. is not a latency bound.
BA, BFT, SMR	Byzantine agreement, Byzantine fault tolerance, and state machine replication. MBA names the source's construction with agreement per vector position.
LMD-GHOST; Votor	Latest-message-driven GHOST, the fork-choice rule named in BRAID's parallel tracks; the Alpenglow consensus component inherited by Constellation.
ACS; $\mathcal{C}_{\mathrm{ACS}}(b)$	Asynchronous common subset. The shorthand is $O(n^2 b + \lambda n^3 \log n)$ bits per ACS epoch, with maximum input b bits per validator and security parameter λ.
DA; DA-CR; DA-vote	Data availability; BigDipper's data-availability and censorship-resistance component; a vote attesting to the specified availability condition.
AVID-M; RB; QC	DispersedLedger's verifiable information-dispersal mechanism; reliable broadcast; quorum certificate. The mechanism or certificate does not by itself imply a recovery deadline.
GNR; w.o.p.	Garimidi–Neu–Resnick; with overwhelming probability, meaning a failure probability negligible in the security parameter under the source model.
tx, req, msgs, sigs	Transactions, requests, messages, signatures. "/tx" is per transaction. "Amort." means amortized.
dedup; k after a number	Deduplication: processing a repeated transaction identifier once. In rates such as 20k tx/s, k means one thousand, not a protocol parameter.

Table 5 (continued)

Symbol or label	Meaning and scope
O, E, model in rate cells	Ordered/delivered work, a benchmark including execution, and an analytical capacity model. These are not measurements of the same endpoint.
$O(\cdot), \Theta(\cdot)$	Asymptotic upper and matching-order bounds, retaining the cell's units and assumptions. The standalone O in a rate cell is instead the ordering/delivery label.
δ, Δ, R_D	Actual message delay, the post-GST delivery bound, and a delivered DAG round. A DAG round may require several network exchanges.
k in BRAID	Number of concurrent proposer tracks. The naive payload multiplier counts full block data; the bribery claim assumes the talk's economic model.
b, λ in BigDipper	Per-validator input (mini-block) size and cryptographic/security parameter in the source's communication units. C_{band} is one validator's bandwidth; $O(nC_{\text{band}})$ is a capacity model, not executed goodput.
m in AMP	Payload size in the byte-cost expression $O((m+n)n^2 \log n)$; the analysis also assumes constant-size identifiers.
K in Gatling	Number of interleaved base-consensus instances. It is not a validator count or a complete communication bound.
c, κ in Strong Prefix	Proposal-payload size and cryptographic size in bits; the displayed $O(cn^2 + \kappa n^3)$ cost is for the good case.
b, λ in DispersedLedger	Maximum current proposer-block size and cryptographic/security parameter in the per-validator, per-epoch cost; linked recovery traffic is not included in the reported bound.
q, λ in Constellation	Number of attesters and length of the rolling cycle window. Neither is the q of Table 1 or a cryptographic λ. A $q/4$ threshold means that many of the q attesters' coded fragments.
$\hat{m}, \lambda$ in Hermes	Number of lanes and cryptographic size; the source represents a tipcut (lane-endpoint vector) as size $O(\hat{m})$ in its notation.
e in Multimmit	Maximum extension in transaction-block heights beyond the preceding safe lane tip, as specified by its inclusion theorem.
α, η, k in PoW rows	Adversarial hash-power fraction; GHOSTDAG's security margin; its bound on the number of other cluster blocks incomparable with a given block. This k is not a proposer count.
ε, λ_b	Target confirmation-error probability and mining/block-arrival rate in the applicable proof-of-work rows.
C_h in Prism	Modeled honest capacity $(1-\alpha)C$, where C is the network-capacity parameter.

Table 5 (continued)

Symbol or label	Meaning and scope
k, p, n, L in OHIE	Number of chains, success probability per mining query, number of miners, and $L = \log(1/\varepsilon) + \log k$. Its cadence uses model query ticks, not seconds.
PoW; EC; EVM	Proof of work; Filecoin's Expected Consensus; Ethereum Virtual Machine. A Filecoin null epoch has no block on the adopted chain.
KB, KiB, MB, Mbps	Kilobytes, kibibytes (1024 bytes), megabytes, and megabits per second; KB and MB retain the source's notation. Ggas/s means billions of gas units per second. A gas rate is not a transaction count.

E.3 Architecture and scope

Table 6. Five comparison classes and the usual scope of their central instances. The four direct MCP rows are architectural families. Constrained assembly is an MCP-adjacent comparison class. Missing proposal handling and selected body recovery are separate policies.

Comparison class	Central scope	Decides	Proposal organization	Examples
Constrained assembly	MCP adjacent	One checked assembly	Per step constraints	FOCIL, AUCIL [105,109]
Fixed proposal vector	Direct MCP	Labeled entries or prefix	Fresh per step positions	Cadence, BigDipper-7 [8,113]
Selected proposal collection	Direct MCP	Subset, supported prefix, or certificate map	Fresh candidates without assigned positions	Multiplicity, Raptr, HoneyBadgerBFT [38,107,71]
Persistent proposal streams	Direct MCP	Cut, prefixes, or interleaving	Growing lanes or instances	Autobahn, DispersedLedger, Gatling [42,114,92]
Causal proposal DAG	Direct MCP	Anchor and causal past	Cross author references	Bullshark, Mysticeti [100,7]

E.4 Guarantee evidence matrices

Reading convention. Each cell states the actual service, premise, or endpoint. The tables assign no ordinal grades. The main-text colors distinguish evidence types;

they do not replace the actual claims or their qualifications. Italic entries record evidence status or provenance rather than service strength. In particular, *Inherited* identifies a claim supplied by a named base protocol, *Unspecified* identifies a missing mechanism or composition rule, and *N/A* identifies an endpoint the component does not supply. Stars point to qualifications below the relevant panel.

Table 7. Guarantees for the motivating configurations. Sans serif tags record reconciliation class and italic tags record indirect scope.

Protocol	Safety	Liveness	Inclusion	Hiding	Last-look protection
Multiplicity [SC] [38]	*Inherited from base protocol*	*Inherited from base protocol*	Tx in $> 1/3$ bundles mandatory*	Plaintext	No; bundle choice*
FOCIL [CA] *[adjacent]* [105]	*Inherited from base protocol*	Base protocol with list validity	Listed tx in next slot*	Plaintext	No; assembly*
BRAID [FV] [87]	*LMD-GHOST; finality overlay**	Slowest track*	Economic bribery argument*	Sealing unspecified*	Sealing / release unspecified*
MBA [FV] [88]	*Inherited from base protocol*	Slot coupling*	Entry validity / admission*	Optional overlay*	Overlay dependent*
Cadence [FV] [8]	$3f < n$	PSync with clocks*	Same slot*	Cutoff for new bodies*	New proposals*
GNR + coded hiding [SC] [41]	$5f < n$	PSync, w.o.p.*	Include-or-empty*	Hidden across proposers until valency*	Public-signal exposure*
Supernova Part One [SC] [88]	Integration unspecified*	Recovery unspecified*	Receipt rule*	Hiding unspecified*	Binding rule unspecified*
Constellation [SC] [5]	*Inherited from Votor**	*Inherited from Votor**	Include-or-empty window*	Plaintext	No; proposal inputs*

Notes to Table 7. Multiplicity. A valid leader assembly selects at least two thirds of the authenticated bundles, which makes a transaction carried by more than one

third mandatory in every such selected collection. Block liveness of the base protocol remains separate. When several qualifying collections exist, the assembler can observe their signed plaintext bundles and then select among them. The threshold therefore does not bind bundle selection before later information is available. **FOCIL.** The cell starts only once a transaction is carried on an eligible inclusion list without equivocation that reaches the proposer and a set of honest attesters relevant to fork choice before their freeze deadlines. The supported block in slot $N + 1$ must then contain it unless it is already present, invalid against the state after the payload, or cannot fit in the remaining gas. List construction is discretionary, so receipt by one committee member does not establish service in the next slot. The rule also stops before state finality. The builder receives lists before completing its payload and may place listed transactions anywhere in the block. It can therefore observe a listed bid, construct its own response, and choose its placement while satisfying the list rule. Assembly remains exposed to last look [105]. **BRAID / MBA.** BRAID's cited talk merges same-slot blocks from parallel LMD-GHOST tracks, deduplicates transactions, and applies a fee-based order. The talk attributes eventual consistency to the slowest track and adds a Byzantine-agreement overlay for finality. Its censorship argument requires bribing all relevant proposers; it is not a receipt-to-inclusion deadline. Proposed sealing mechanisms lack a specified observer/release guarantee in this source. The talk is a design description, not a complete fault or service proof. MBA resolves vector positions using its base agreement protocol. "Position coupling" means that the vector cannot close while a required position remains unresolved; an agreed empty entry or fallback must be supplied by the configuration. An "overlay" is an added mechanism, not a guarantee of the underlying agreement. MBA's transaction service depends on the entry-validity and admission rules; vector agreement alone supplies neither a protected transaction nor a merge policy. The proposed sealing and optional overlays do not specify a complete observer, binding, and release contract, so their hiding and last-look entries state this dependency rather than crediting a guarantee. Ethereum's two-proposer proposal remains a separate row: its joint fork choice and missing-payload rules are not BRAID's track rules.

Cadence. Static corruption, synchronized clocks, and partial synchrony are required. After the specified grace period following GST, timely proposals from correct designated proposers enter their slot, including when fallback is used. A split that blocks a quorum can trigger fallback for the whole slot. Threshold encryption restricts new reactive proposals, but does not prevent choosing among previously eligible faulty candidates. Appendix B.6 explains the failure amplification and accountability assumptions. Section 4.4 establishes body recovery in both fast and fallback paths, with deterministic rejection of malformed encodings and threshold opening. The hiding cutoff is the slot deadline when correct validators release shares. **GNR.** The model assumes static corruption, synchronized clocks, private links between nodes with signed protocol messages, partial synchrony, and a secure base atomic broadcast protocol at $f < n/5$ with liveness driven by a leader. Relay sampling gives progress with overwhelming probability, selective censorship resistance, and hiding across proposers. Bad samples can stall without violating agreement. The service claim applies separately to each slot under an include-or-empty rule. A timely honest submission enters a nonempty slot, but a globally empty slot is permitted and the construction supplies no separate bound on consecutive empty slots. The cell therefore records qualified omission that cannot target individual transactions rather than bounded transaction service. The hiding game protects a transaction submitted to an honest receiving node from the other protocol nodes until it enters protocol valency. It does not hide the transaction from that receiving node or equalize actors' submission times. GNR explicitly retains

the residual last-look advantage from a later actor observing a fresher public signal, so no last-look protection is credited. Its reconstruction phase recovers every selected batch from honest relays under the source's sampling and threshold assumptions.
Supernova / Constellation. "Receipt rule" means that attestations about received proposals constrain what the selector may omit. Supernova Part One is recorded at the design-description level of the cited presentation; the row does not infer a complete fault, recovery, or service theorem from that description. In particular, its integration with base consensus, recovery path, hiding and binding mechanisms, and transaction merge policy are not sufficiently specified in the cited evidence. The receipt rule constrains selection, not transaction sequencing. Constellation supplies the more explicit rule below. Constellation's v0.9 specification assumes less than 20% Byzantine stake and up to a further 20% crashed stake, partial synchrony, accurate clocks, and the stated committee-sampling conditions. Votor is the consensus component of Alpenglow; the inherited fault assumptions are not a separate MCP result [54,55]. The extra crashed stake is not an additional Byzantine allowance. Constellation accepts an attestation aggregate only if at least 60% of the q attesters contribute to it, and the leader must include every valid proposal slice attested by at least 40% of them, with fraud-witness handling for malformed submissions. Its selective-censorship property states that a correct proposer's transaction submitted in cycle c is included in some cycle $c' \leq c + \lambda$, or no batches appear from such a c' through $c + \lambda$. This permits an empty suffix; it is not unconditional service within λ cycles. Execution additionally requires its fee-competitiveness condition. Fourfold coding permits recovery from $q/4$ fragments at the leader. The leader includes reconstructed data in full, and Rotor disseminates it to validators, who validate the block before consensus. Availability of committed data therefore uses this complete path, not an inference that $q/4$ attesters alone guarantee $q/4$ honest holders. Committee sampling, clocks, Rotor, and Votor remain the named composition assumptions.

Notes to Table 8. AUCIL. Fees and equilibrium incentives supplement the base protocol's inclusion mechanism. They are not a Byzantine inclusion bound. FairFIL's accountability and bribery analysis likewise does not turn client receipt into an eligible list premise enforced by the protocol [101]. **BigDipper-7.** This row is the integrated Navigator, Card-7, Scooping, and HotStuff-2 composition. The fixed vector tag follows from the preassigned logical validator positions resolved by the committed object, not merely from the codeword representation. Card-7 without the committing integration remains a data availability component. The integration preserves the base protocol's safety and liveness under partial synchrony. The progress claim also uses $3f < n$. Corruption adaptivity is not specified. After GST an honest leader completes DA-CR, while a faulty leader is handled by timeout and view change. A completed Card-7 instance contains at least $t = n - 2f$ honest mini blocks. An honest client that obtains positive responses for the target view from $x \geq n - t + 1 = 2f + 1$ distinct available validators before mini block formation is included in every completed instance. If a leader attempts to omit a transaction meeting this premise, it cannot complete the DA-CR certificate, and honest validators accept no block for that view. This is certified data inclusion, not execution. Navigator sends plaintext, so no hiding guarantee is credited. The configuration specifies no protected last-look action or theorem, and residual leader selection and mini block grinding remain, so no last-look guarantee is credited. A completed certificate provides eventual retrievability and uniqueness. DA-CR supplies eventual full or partial retrieval outside the main path. Ordering within the object, application ordering, execution, and serving remain duties of higher layers.
AMP. If all correct validators attest a proposal at height h, it is selected at $h+1$, under

Table 8. Guarantees for inclusion mechanisms and selected collection protocols.

Protocol	Safety	Liveness	Inclusion	Hiding	Last-look protection
AUCIL [CA] *[adjacent]* [109]	*Inherited from base protocol*	*Inherited from base protocol*	Economic*	Plaintext	No; assembly*
BigDipper-7 [FV] [113]	$3f < n$*	PSync with eventual honest leader*	Data inclusion in each view or no block*	Plaintext	No; residual selection*
AMP [SC] [21]	$3f < n$	PSync	Next height*	Plaintext	No; proposal inputs*
Ethereum (two proposers) [FV] [78]	*Inherited from base protocol*	One missing payload permitted*	Twice the bribery cost; model*	Plaintext	No; payload choice*
HoneyBadgerBFT [SC] [71]	$3f < n$	Async	Backlog bound*	After selection	Blind selection
Dumbo2 (ACS) [SC] [44,43]	$3f < n$	Async	Aggregate only*	Plaintext ACS*	No; ACS inputs*
Red Belly [SC] [28]	$3f < n$	PSync	Eventual; no conflicts*	Plaintext	No; proposal inputs*

the protocol's timing and validity premises. Section 4.3 and Appendix A, Lemma 4, establish eventual payload retrieval: each selected identifier has more than f attestations and hence a correct holder. Deterministic payload merging supplies the order; the exact policy is left to the application. **Ethereum (two proposers).** Section 2.1 permits the available payload to become canonical when the other is missing. The post's bribery model doubles the cost of buying both proposers' omission, assuming the specified incentives; it gives no unconditional transaction deadline. Section 3 keeps the attestation count unchanged while doubling payloads. Attestation size, payload replication, and fork-choice behavior remain separate costs and assumptions. **HoneyBadgerBFT.** Threshold decryption follows common-subset selection, so participants select ciphertexts before their contents are revealed. This is the "blind selection" entry; it does not establish protection against every later public-signal or release decision. For a transaction received by at least $n - f$ correct nodes, the paper bounds inclusion by $O(T/B + \lambda)$ epochs except with negligible probability. Here, T is the pending backlog, B the epoch batch parameter, and λ the security parameter. The adversary is static. **Dumbo2.** This row is the asynchronous common subset (ACS) configuration. Selecting $n - f$ inputs is an aggregate guarantee under static faults. An encrypted atomic broadcast conversion can add hiding, but is a different configuration. **Red Belly.** Requests must reach $f + 1$ proposers, with $2f < |P|$ for proposer set P, and be admissible, nonconflicting, and below capacity. Its reliable-broadcast layer delivers selected proposal bodies, so availability is established. The inclusion claim protects nonconflicting requests under the stated routing and admission premises; it does not require including every conflicting submission in the final superblock.

Notes to Table 9. Gatling. The compared configuration interleaves K Simplex instances and inherits their fault threshold and payload delivery. Definition 2 and the liveness proof in Appendix A establish eventual transaction inclusion when every correct node inputs the transaction; Algorithm 1 forwards it to all component instances. Section 6 explicitly distinguishes this premise from receipt by just one correct node. Availability follows the Simplex payload delivery path; Gatling merges the delivered logs. "K base instances" is a description of structure, not a byte bound; timing and workload notes appear in Table 14.

Mir-BFT / ISS. Rotating transaction buckets and retransmission support eventual service. Mir's totality and liveness properties and ISS's total-order broadcast interface concern delivered requests, so their named broadcast and recovery mechanisms also supply availability. A fixed delay requires the configuration's rotation and recovery parameters. **Ladon.** Ladon claims safety for $3f < n$. Gatling's analysis of related work reports a counterexample to its global ordering rule. The disputed guarantee must be resolved before using Ladon as a safe baseline. The underlying sequenced broadcast uses Byzantine reliable broadcast to deliver block bodies; this supplies availability independently of the disputed global merge. Theorem 5 claims eventual inclusion, but no safe canonical-inclusion result is credited while the global ordering guarantee remains disputed. **Orthrus.** Safety is for its execution model based on objects. Lemma 6 and Theorem 2 establish eventual confirmation of a correct client's transaction under the routing and bucket-rotation rules. Confirmation may record an abort, so it does not guarantee successful execution. The sequenced broadcast layer delivers transaction bodies to correct validators; this supplies availability before the partial or global ordering path processes them. **Raptr.** The configuration commits supported prefixes of a leader's ordered list of independently authored batches. Theorem C.13 establishes atomic-broadcast validity and totality: an honest validator's broadcast batch eventually reaches every honest validator. Leader discretion in one proposal therefore does not

Table 9. Guarantees for persistent streams, the Strong Prefix fixed vector configuration, and the Raptr selected collection configuration.

Protocol	Safety	Liveness	Inclusion	Hiding	Last-look protection
Gatling (Simplex) [PS] [92]	$3f < n$	PSync	Eventual; all correct input*	Plaintext	No; proposal inputs*
Mir-BFT [PS] [102]	$3f < n$	PSync	Eventual*	Plaintext	No; proposal inputs*
ISS [PS] [103]	$3f < n$	PSync	Eventual*	Plaintext	No; proposal inputs*
Ladon [PS] [63,92]	Disputed*	PSync*	Eventual claimed; order disputed*	Plaintext	No; proposal inputs*
Orthrus [PS] [62]	$3f < n$ for objects*	PSync	Eventual; may abort*	Plaintext	No; proposal inputs*
Raptr [SC] [107]	$3f < n$	PSync	Eventual*	Plaintext	No; leader list*
Strong Prefix [FV] [112]	$3f < n$	PSync	Bound of f slots*	Plaintext	No; input choice*
DispersedLedger [PS] [114]	$3f < n$	Async	Every correct block, eventual*	Plaintext	No; input choice*
Autobahn [PS] [42]	$3f < n$	PSync	Eventual*	Plaintext	No; fresh cuts*
Sei Giga [PS] [68]	Autobahn ordering with $3f < n$*	PSync*	Eventual*	Sedna proposed; bounded leakage*	Content protection proposed; public signals remain*
Hermes (v2, Autobahn lanes) [PS] [85]	$5f < n$	PSync*	Eventual*	Plaintext	No; fresh cuts / skips*
Multimmit (v4) [PS] [59]	$5f < n$	PSync	Eventual*	Plaintext	No; lane inputs*

justify withholding eventual inclusion from this row. Quorum Store and the prefix-support threshold guarantee a correct holder for committed batches; retrieval supplies availability. Admission of a client's transaction to such a broadcast batch remains the input premise.

Strong Prefix. A censored slot is one that omits any honest party's input for that slot. Definition 2.7 and Theorem B.5 bound the total number of such post-GST slots by f; this is not a fresh allowance of f omissions in every window. Section 6.1.5 and Lemma A.2 establish availability of decided hashes from correct holders, with a dispersal layer that retains and serves their preimages. The bound protects slot inputs, so client service also needs admission or resubmission through an honest proposer. The displayed 4δ is an amortized commit latency in the failure-free synchronized case; the agreed high output closes the full slot later [112]. **DispersedLedger.** The configuration has fixed n, authenticated channels, and randomized asynchronous binary agreement. Agreement and the next epoch's dispersal phase can proceed before each validator retrieves all bodies from earlier epochs. Its eventual service theorem retains every correct proposer's block through completion prefix observations for each author in links between nodes and ultimately delivers it to all correct validators. At the client endpoint, the premise is submission from a correct client to a correct server. The result is eventual transaction delivery and execution, not a finite asynchronous horizon. AVID-M coding establishes availability rather than semantic hiding, and the configuration states no binding point that closes every fresh input before other inputs become visible. AVID-M's retrieval handler releases chunks once dispersal completes, without waiting for the epoch's agreement outcome. An asynchronous execution can therefore deliver and reconstruct one proposal while delaying agreement messages, allowing a Byzantine uploader to form a responsive proposal before that epoch closes (Figure 4 of [114]). After at least $n - f$ binary agreements output one, a correct node inputs zero to each remaining binary agreement instance on which it has not yet invoked input. A completed but malformed or inconsistent AVID-M body resolves through the protocol's deterministic bad uploader or malformed input handling rather than blocking. Correct omitted blocks remain eligible for later links between nodes.

Autobahn. Theorem 2 proves eventual ordering and execution of a client request resent until it reaches a correct validator. Lemmas 11–13 show admission to that validator's lane and eventual commitment of every correct lane block. Correct leaders propose the highest known tips; the proof assumes continuing lane production sufficient for its coverage rule. A stalled lane need not stop other lanes. Under static faults, an $f+1$-vote availability certificate gives at least one correct holder; Lemma 10 proves retrieval of a committed cut's entire referenced data. Fewer guaranteed holders than Narwhal is a redundancy difference, not absence of availability. No adaptive-custody or fixed client deadline is inferred. Section 5.2.3 leaves lane coverage unenforced for Byzantine leaders. Before a fresh cut is fixed, such a leader can observe lane contents and choose among valid frontiers, including advancing one eligible lane while retaining an older head for another. This gives a last-look choice over the current cut. Eventual inclusion by a later correct leader does not remove that discretion [42]. **Sei Giga.** The base Autobahn ordering layer inherits $3f < n$ safety, partial-synchrony progress, and eventual inclusion through committed ordering. In version 3, Proposition 3.1 gives a correct holder for an $f + 1$-vote availability certificate (PoA); Assumption 3.2 requires retention and serving until all correct validators retrieve the data; Corollary 3.3 establishes that retrieval for every committed cut. Thus availability is guaranteed under the stated retention model. Deterministic execution and unique $2f + 1$ state-digest attestations have additional state and signing premises; no bounded submission-to-state-finality

theorem is claimed. The hiding and last-look cells credit the proposed Sedna path in Section 5. It commits to payloads and distributes coded symbols; honest executors reconstruct after enough symbols are finalized. This proposes content protection until decode. The cited coded construction bounds leakage from symbols and headers rather than guaranteeing full semantic hiding. It uses private coded dissemination rather than threshold encryption. Section 5.3 nevertheless warns that a withholding coalition can gain an informational lead. Protection against content-dependent responses within the same cut therefore requires hiding to last until the relevant lane and cut choices bind. Hiding does not close responses to later public prices [68,86]. For the base plaintext path, Definition 4.1 sorts lanes by their maximum transaction tip, preserves intra-lane order, and removes duplicates. An honest lane can expose a fixed transaction x before cut selection. A Byzantine proposer can then certify a fresh lane block containing a response y. If both lanes enter that cut and the response lane has the higher maximum tip, its transactions precede x. A Byzantine cut leader also retains Autobahn's fresh-cut discretion. This deduction establishes exposure in the plaintext path; it is not a counterexample to an instantiation that hides x until those choices bind [68,42].
Hermes. The row evaluates version 2's multi-lane construction with Autobahn-style data availability (Section 4.2). Its $5f < n$ threshold remains different from Autobahn's $3f < n$. Lemmas 8–9 guarantee availability transfer and reconstruction of the consensus values; selected transaction bodies rely on the named lane layer's storage and retrieval obligations. Eventual inclusion follows compositionally: a valid transaction admitted to a correct lane eventually reaches every correct process with its ancestry. Section 4.2.3's GetChild includes every consecutive locally held block above the parent's height frontier. A later honest leader therefore proposes the block unless its parent already contains it, and Lemma 7 finalizes an extension of that proposal. Skips advance positions without discarding lane blocks or advancing their height frontier (Section 4.2.2). This argument uses the prescribed cut construction, correct-lane admission and retention, and eventual honest leadership; it is not a theorem for arbitrary lane policies or for every available Byzantine block. The same-view honest-leader guarantee remains stronger than eventual service. The 2δ latency starts with a ready view and excludes prior lane dissemination. For last look, the honest GetChild rule is not a validity requirement to include every locally known block. Section 4.2.4 checks the required parent, frontiers, ancestry, and prefix extension. Subject to those checks, a Byzantine preferred leader can choose a fresh valid cut or skip after observing plaintext or a later public signal. Previously fixed prefixes remain binding; the exposure concerns the fresh extension, not revision of a committed prefix. **Multimmit.** Version 4, Theorem 2, proves that a transaction received by a correct processor eventually enters every correct processor's log. This supports the shared eventual-inclusion label. Lemma 12 proves full-data availability for emitted blocks and their ancestry: DA-certified blocks have at least $2f + 1$ correct holders, while a tip represented by a view certificate can have only one, which still suffices for retrieval under the model. Theorem 3 additionally protects a correct producer's block when every correct processor has DA-voted it and its required ancestry before its view vote, within e heights of the preceding safe tip. If that leader block finalizes, membership is fixed; the next finalized leader orders it, or the good case orders it immediately. A leader can prevent its own view from finalizing, so this is not an unconditional next-view deadline. These stronger conditions differ from Hermes' honest-leader proposal guarantee. The negative last-look entry concerns fresh lane inputs. A producer may observe a fixed plaintext input before choosing its own next block. Theorem 3 can force qualified block membership once its voting premises hold,

but does not bind all producer inputs before that earlier observation. We do not infer arbitrary omission of blocks protected by that theorem.

Table 10. Guarantees for complete causal DAG configurations.

Protocol	Safety	Liveness	Inclusion	Hiding	Last-look protection
Narwhal–HotStuff [CD] [31]	$3f < n$	PSync	Eventual*	Plaintext	No; vertex inputs*
Tusk [CD] [31,93]	$3f < n$	Async claim refuted*	Fails in analyzed attack*	Plaintext	No; vertex inputs*
Bullshark (PSync) [CD] [100]	$3f < n$	PSync	Eventual*	Plaintext	No; vertex inputs*
DAG-Rider [CD] [50]	$3f < n$	Async	Eventual*	Plaintext	No; vertex inputs*
Mysticeti-C [CD] [7]	$3f < n$	PSync	Eventual*	Plaintext	No; vertex inputs*
Fino [CD] [65]	$3f < n$	PSync*	Eventual*	After ordering	Blind ordering

Notes to Table 10. Narwhal–HotStuff. This is the composition in Sections 3.2–3.3 of the Narwhal paper, not standalone Narwhal. HotStuff supplies ledger safety and partial-synchrony progress at $3f < n$; committed certificates select their uncommitted causal histories for deterministic ordering. Narwhal's Block-Availability property and Lemma A.2 guarantee recovery under eventually reliable links. Its $2f + 1$ certificate contains at least $f + 1$ correct full-data holders. Section 3.3 requires reinjecting transactions from uncommitted garbage-collected blocks; with the composition's progress and honest admission, this supplies eventual transaction inclusion. It does not guarantee that every original DAG block is committed. Standalone Narwhal needs no partial-synchrony assumption for data delivery, but supplies no ledger order on its own.

Tusk. Narwhal and Tusk claim asynchronous liveness for the construction. Shoup gives an adversary that controls message scheduling and learns each wave's coin from the f corrupt shares; for sufficiently large n it drives all parties through arbitrarily many waves while only one honest party ever commits a leader. The attack uses the published specification's lazy commit rule and a coin with reconstruction threshold $f + 1$ whose shares are revealed after the second round of a wave, readings the Tusk authors confirmed to Shoup. An aggressive commit rule with back-pressure leaves only a weaker liveness, and a high-threshold coin appears to defeat this attack but has no known liveness proof. The counterexample concerns this construction, not asynchronous DAG protocols generally, and Shoup knows of no production deployment it affects [93]. Its Narwhal data layer still guarantees availability of certified bodies; the counterexample does not refute that separate property. In the analyzed lazy-commit execution, honest parties other than the distinguished committer never deliver the leaders, despite eventual delivery of the DAG vertices. Common canonical inclusion

therefore fails. The aggressive-commit variant is distinct and can restore eventual commitment under eventual delivery while allowing unbounded wave lag. **Bullshark / DAG-Rider / Mysticeti.** Eventual inclusion concerns eligible correctly disseminated contributions. None of these entries claims a client delay independent of workload. The Bullshark row is its standalone partially synchronous construction. DAG-Rider assumes a suitable shared coin. Mysticeti's cited analysis uses a static adversary within an epoch. Bullshark inherits Narwhal recovery; DAG-Rider uses reliable broadcast; Mysticeti-C establishes delivery through its DAG and atomic-broadcast agreement (Theorem 3), without explicit per-block certificates. Lifefin shows that unbounded growth in pending transactions can prevent commitment without its ACS fallback and recovery overlay with bounded resources. The matrix does not infer those premises from Mysticeti or DAG structure [118]. **Fino.** Ordering precedes threshold opening (decryption), so the ordering decision is made without access to the protected plaintext. Eventual reconstruction and the opening assumptions are required before execution. Its ordering and opening delays are reported separately in the performance table. The DAG transport's reliable delivery makes selected bodies and ancestry recoverable; Lemma 6.4 gives unique decryption or agreed rejection of committed inputs. Eventual inclusion of an honest disseminated input follows from eventual delivery to honest leaders and commitment of their causal histories after GST (Sections 4–6.2). This is not a bounded client-service claim.

Table 11. Guarantees for boundary comparisons with probabilistic finality. Class tags describe reconciliation structure and do not change the panel's boundary scope.

Protocol	Safety	Liveness	Inclusion	Hiding	Last-look protection
SPECTRE [CD] [97]	$\alpha < 1/2$ for pairs*	Sync	Published nonconflicting tx: eventual*	Plaintext	No; mining inputs*
GHOSTDAG [CD] [98]	$\alpha < (1-\eta)/2$*	Sync with tuned k	Published blocks: probabilistic*	Plaintext	No; mining inputs*
Conflux [CD] [60]	$\alpha < 1/2$	Sync	Published blocks: probabilistic*	Plaintext	No; mining inputs*
Prism [CD] [9]	$\alpha < 1/2$	Sync with capacity premise*	Tx in a block: probabilistic*	Plaintext	No; mining inputs*
OHIE [PS] [116]	$\alpha < 1/2$	Sync	All honest receive: probabilistic bound*	Plaintext	No; mining inputs*
Filecoin EC [SC] [81,?]	$\beta < \beta_\star$*	Sync; secure beacon*	All honest receive: probabilistic bound*	Plaintext	No; block inputs*

Notes to Table 11. Here α is the adversary's hash-power fraction in the five proof-of-work rows. Filecoin instead uses storage power. Confirmation is probabilistic under each source's network and parameter assumptions. A guarantee starting from a published block protects transactions already admitted to that block; it does not supply a deadline for admission from a client's mempool. Ordered inclusion does not imply successful execution. **SPECTRE.** Weak liveness protects published, nonconflicting transactions whose inputs are robustly accepted. Its pairwise precedence relation does not supply a general total order for arbitrary state machine execution. Conflicting transactions can remain undecided under a balancing attack [97]. **GHOSTDAG.** Property 1 and Theorem 4 establish convergence of the order of published blocks under the stated hash-power bound. The cluster parameter k must match the block rate, propagation delay, and security margin η. Acceptance of a transaction additionally requires that no earlier accepted transaction conflict with it. The result protects published block contents, not admission from a client's mempool [98]. **Conflux.** Section 4.3 establishes that a block received by an honest node obtains a stable position with probability at least $1-\varepsilon$ after $d\,O(\log(1/\varepsilon))$ time under the source's assumptions, where d bounds propagation delay. Its admissible, nonconflicting transactions inherit ordered inclusion; client admission and successful execution remain separate [60]. **Prism.** Once a transaction enters a transaction block, Theorem 4.2 gives reference by a permanent leader-sequence block within $O(\log(1/\varepsilon)/(1-2\alpha)^4)$ rounds with probability at least $1-\varepsilon$. The source's network capacity and parameter premises remain. Its faster confirmation result for honest payments has additional conditions and does not establish equally fast general transaction ordering [9]. **OHIE.** Section V.D derives from Theorem 1 that a transaction continuously supplied to all honest nodes enters the confirmed block sequence within $\Theta((\log(1/\varepsilon)+\log k)\Delta)$ ticks with probability at least $1-\varepsilon$, under the stated mining parameters and honest admission model. Here k counts chains. Membership comes from confirmed prefixes; the trailing-block reference assigns rank, so OHIE remains a persistent stream (Section 3.6) [116]. **Filecoin EC.** Definition 3 and Theorem 4 of [**?**] establish permanent inclusion within κ epochs after receipt by all honest nodes, with probability at least $1-\exp(-\Omega(\kappa^{1-\epsilon}))$ for any fixed $0<\epsilon<1$. The analyzed model assumes static storage power and corruption, synchronous epochs, the specified rebroadcast rule, and a secure random beacon. It requires $\beta<\beta_\star$, where β is adversarial storage share and $\beta_\star$ is the unique solution of $\beta_\star m = 1-e^{-(1-\beta_\star)m}$ for the fixed expected leader count $m>0$. This is a guarantee of the analyzed EC model, not an unconditional fee or capacity guarantee of a deployment. Tipsets are selected collections; native probabilistic confirmation remains distinct from a composed finality mechanism [81,82].

E.5 Performance, latency, and cadence

Tables 12 through 16 report the same 39 configurations as the guarantee tables. *NR* means no verified result for that metric in the cited source. *N/A* means the component supplies no such endpoint. Results are unranked because different workloads, hardware, fault settings, and endpoints preclude a common numerical ranking. Rates marked O report ordered or delivered payload at a named endpoint before execution. The source unit remains transactions, requests, or bytes. E denotes a benchmark that includes execution. Neither automatically measures unique, successful, economically useful work. M marks a modeled capacity, not a benchmark.

Latency starts matter. P denotes proposal dissemination to protocol commit, U client submission to commit, and R submission to client replies. A star identifies a different start or endpoint in the notes. An unmeasured protocol bound is kept separate from measured latency. The actual delay δ and synchrony bound Δ are not interchangeable. R_D denotes a delivered DAG round, which can contain several network exchanges. Expected asynchronous rounds and probabilistic confirmation depths are not fixed bounds in wall clock time. The cadence column records the named production or decision unit and its interval when supplied. It is not the reciprocal of confirmation latency.

Communication is aggregate unless identified as an object size or factor per byte. Message counts remain labeled messages. We write $\mathcal{C}_{\mathrm{ACS}}(b) = O(n^2 b + \lambda n^3 \log n)$ bits for an ACS epoch with maximum input b per validator and security parameter λ. Payload size, batching, proofs, retransmission, and duplication must be included before these costs can become a comparison of useful goodput.

Notes to Table 14 (continued). Strong Prefix's 4δ amortized early commit assumes no failures and synchronized slot starts. Sequential slots still wait 8δ for their agreed high output. With proposal size c and cryptographic size κ, the displayed communication is for the good case. Communication in the worst case is $O(cn^2 + \kappa n^4)$. Its separate primitive with two rounds at $n \geq 5f + 1$ is not the measured or analyzed SMR configuration [112]. DispersedLedger's displayed cost derives from n AVID-M instances and n parallel binary agreements per epoch, with b denoting the maximum current proposer block. Consensus can agree on commitments before body retrieval. Each retained block is retrieved at most once per correct validator, but recovery along a linked prefix can span multiple prior blocks per author. The source gives no aggregate or burst traffic bound for recovery. Each binary agreement has expected constant asynchronous round complexity, but the source does not state a round bound for a full epoch or a bound in wall clock time. A node may propose its next block after 100 ms or 150 KB, whichever comes first, subject to completion of all binary agreements from the prior epoch. In a WAN across 16 cities, the reported mean 36 MB/s per server summarizes throughput of confirmed transactions rather than executed goodput. Median latency from a transaction entering its local node to delivery rises from 730 to 830 ms as offered load rises from 2 to 23 MB/s. The reported p99 is about one second only at the separately tested 6 MB/s load [114]. Autobahn's default path using a certified tip takes $6\delta/8\delta$ from data dissemination to validator commit, versus $3\delta/5\delta$ from cut proposal. Fast paths require all votes. Linear control messages require aggregated signatures, and cut bytes still scale with the lanes. Its prototype's observations at the execution readiness endpoint use optional optimizations and $n = 4$, so they are not measurements of every default path [42]. Sei Giga v3 reports an internal run with 40 validators across 20 regions above 5 Ggas/s with ordering finality below 250 ms. We mark the rate E because it is executed EVM gas, not because it establishes goodput of distinct successful transactions. The source gives no duplicate or conflict adjusted rate, run distribution, tail latency, or latency of state attestation [68]. Hermes v2 uses $\hat{m}$ for the number of lanes, with tipcut size $O(\hat{m})$ in its notation, and λ for cryptographic size. Table 2 gives $O(n^2\hat{m} + \lambda n^3)$ bits per view for amplified multi-lane values, rather than $O(n^3\hat{m})$ with full-value forwarding. This coordinates values; it is not an end-to-end cost for transaction production and execution. Its good-case 2δ starts at synchronized view entry with an available proposal and ends at prefix finalization. From the first honest post-GST entry, Lemma 7 instead gives 3δ to all honest nodes' finalization. The tabulated fallback is $2\Delta + \delta$; Lemma 6 gives $2\Delta + 2\delta$ from the first entry. Exceptional amplification in Corollary 2 can add δ. None includes earlier lane queueing or transaction execution [85]. Multimmit v4, Section 3.6 and footnote 7, gives 2δ from the leader proposal to ordering in its synchronous all-correct good case. From transaction block dissemination it gives mean 3δ and best 2δ, including the wait for a suitable voting window and requiring the block's freshness. The mean is analytical under the stated arrival model, not a measurement. These clocks must not be ranked as a Hermes 2δ versus Multimmit 3δ comparison. Multimmit's $O(n)$ counts payload copies before recovery. Section 3.7 estimates, for $n = 100$, $K = 100$ chains, $d = 3$, $e = 2$, about 23 KB per proposal and 30 KB per good-case quorum certificate (QC), rising to roughly 0.5 MB per QC under worst-case vote disagreement. These are individual object sizes, not aggregate per-view traffic [59].

Table 12. Performance of headline configurations. Bounds and observations are specific to each source. No row is ranked by an unmatched experiment.

Protocol	**Comm.**	**Protocol bound** start to endpoint	**Cadence unit / interval**	**Source rate**	**Measured latency**
Multiplicity [38]	Committee bundles, byte total NR	Base-protocol finality	Base-protocol slot	NR	NR
FOCIL [105]	8 KiB/list cap	Base-protocol finality	12 s base-protocol slot	NR	NR
BRAID [87]	Naive $k\times$ blocks; batched votes*	Slowest track; bound NR	Concurrent proposal slot	NR	NR
MBA [88]	BA per vector position	NR	Proposal vector step	NR	NR
Cadence [8]	Coded payloads, total NR	2δ after deadline D^*	100 ms simulated slot	NR	219 ms mean*
GNR + coded hiding [41]	$N/K = 5$ code expansion*	Base broadcast + reveal	Atomic broadcast slot	NR	NR
Supernova Part One [88]	NR	NR	Selected collection step	NR	NR
Constellation [5]	29.1 KiB attestation object/batch*	NR	50 ms economic cycle	NR	NR

FOCIL's draft cap covers transaction list content, not replicated network traffic. Lists constrain the following slot, whose finality still comes from Ethereum. Cadence's fast path finalizes by absolute time $D + 2\delta$, namely two voting delays after the slot deadline D. This is protocol delay after the deadline, not latency from transaction submission or from the slot's defined starting time $D - \Delta$. From that starting time the bound is $\Delta + 2\delta$. The simulation's 219 ms is mean slot finalization. Adding the modeled average 50 ms wait to enter a proposal gives 269 ms to ordering finality, not to execution or state finality. The paper reports no empirical tail percentile. The fallback bound does not establish p99 latency [8]. Provided participation is synchronized within Δ and all correct validators start by time t, every correct validator finalizes by $\max(t, \mathrm{GST}) + 5\Delta + \ell_{\mathrm{MVBA}}$. GNR's factor counts only payload symbols from its hiding code, under its $f < n/5$ parameterization. Commitments, masking material, attestations, and reveal traffic are additional. Its choice of atomic broadcast prevents one universal value for added delay. Several Constellation economic cycles feed a consensus slot. The 50 ms interval is not user finality. Its 29.1 KiB estimate is attestation metadata per batch for the source's parameters ($p = 16$, $q = 256$, $\mu = 4$, $\lambda = 6$), with faulty-proposer commitments omitted; it excludes transaction data and replicated traffic. Adjacent batches can reuse metadata. BRAID's talk describes linear growth in block traffic with k tracks and batching of votes, not a complete byte bound.

Table 13. Performance of inclusion mechanisms and selected collection protocols.

Protocol	**Comm.**	**Protocol bound** start to endpoint	**Cadence unit / interval**	**Source rate**	**Measured latency**
AUCIL [109]	NR	Base-protocol finality	Base-protocol IL instance	NR	NR
BigDipper-7 [113]	DA $O(b + \lambda n)$ per validator, BFT $O(\lambda n)$ per validator*	4 optimistic steps in one direction*	One DA-CR / HotStuff-2 view*	$O(nC_{\text{band}})$ capacity in consensus bits M*	NR
AMP [21]	$O((m + n)n^2 \log n)$ bytes	5 gossip rounds*	Tendermint height	NR	NR
Ethereum (two proposers) [78]	Two payloads; unchanged attestation count	Base-protocol finality	12 s base-protocol slot	NR	NR
HoneyBadger BFT [71]	$\mathcal{C}_{\text{ACS}}(b)$ and $O(n)$/tx*	Expected $O(\log n)$ Async rounds*	ACS + decryption epoch	$>$ 20k tx/s O*	NR
Dumbo2 (ACS) [44,43]	$\mathcal{C}_{\text{ACS}}(b)$	Expected $O(1)$ Async rounds	ACS epoch	NR	NR
Red Belly [28]	$O(n^4)$ bits/instance*	NR	Superblock instance	30,684 tx/s E*	3 s*

BigDipper-7's source table decomposes DA and BFT communication per validator as shown and reports four optimistic communication steps in one direction. Its source table also lists an $O(n)$ asymptotic in the block interval without defining a cadence endpoint in wall clock time. The analysis separately derives $O(nC_{\text{band}})$ capacity in consensus bits relative to a validator's bandwidth C_{band}. These are modeled internal endpoints, not overall throughput. One DA-CR instance runs per HotStuff-2 view. Its four Card messages are piggybacked on HotStuff-2's schedule of four messages, so this is not an additional round or latency from submission to execution. The `reach-proc-time` collection wait is configurable but unquantified. Separately, a HotStuff-2 path outside the optimistic case without a double certificate can incur a 3Δ wait on entering a view. Neither quantity measures ordering or the delay from submission to execution. The communication and rate cells cover the Card-7/HotStuff-2 core on the critical path and exclude Navigator's client fanout and retrieval or reconstruction of full bodies outside the critical path. No system benchmark in wall clock time or result for executed goodput is reported. The scaling analysis assumes an active leader with sufficient compute and bandwidth to collect and encode the aggregate. The source notes that performance may be downgraded when the selected leader lacks that capacity. AMP assumes n proposers, payload size m, identifiers of constant size and gossip taking $O(\log n)$ message delays per round. Its five rounds start with payload dissemination and end at finalization, not execution. HoneyBadger's $O(n)$ aggregate bits/transaction requires sufficiently large batches and full queues. Threshold decryption adds a round. Its $>$ 20k rate uses up to 40 nodes. The benchmark counts consensus completion and abstracts execution. Dumbo2 is the ACS baseline, so no decryption or rate for full SMR is silently included. Red Belly's bound in the worst case is per consensus instance. The bound under good execution is $O(n^3)$ bits per committed transaction batch. Its experiment with 1,000 VMs uses 160 consensus machines, with state updates and transaction validation. The other VMs host clients or maintain state. The reported latency of three seconds belongs to that workload.

Table 14. Performance of parallel instances and designs with prefixes.

Protocol	Comm.	Protocol bound start to endpoint	Cadence unit / interval	Source rate	Measured latency
Gatling (Simplex) [92]	K component instances	3Δ, P*	$2\Delta/K$ proposal interval	NR	~ 215 ms*
Mir-BFT [102]	$O(n^2)$ messages	NR	Parallel batches across epochs	$>$ 60k req/s O	NR
ISS [103]	Depends on broadcast	NR	Segments across epochs	$>$ 11.4k req/s O*	NR
Ladon [63,92]	$O(n^2)$ messages*	NR	Ranked partial commits	NR	NR
Orthrus [62]	NR	Payment or contract path	Partial and global ordering	NR	NR
Raptr [107]	$O(n^2)$ control msgs	3δ, P, with mean 5δ*	2δ proposal interval	250k tx/s O	755 ms*
Strong Prefix [112]	$O(cn^2 + \kappa n^3)$ bits*	Amortized 4δ*	8δ full slot*	NR	NR
Dispersed Ledger [114]	Per validator and epoch dispersal/BA $O(b + \lambda n^2)$* recovery NR	Async eventual, full bound NR*	Sequential epochs with a 100 ms / 150 KB proposal gate*	Mean 36 MB/s per server O*	Median 730 to 830 ms*
Autobahn [42]	$O(n)$ control msgs*	$3\delta/5\delta$, P*	Independent cars and cut slots	$\sim$ 234k tx/s O*	~ 280 ms*
Sei Giga [68]	NR	3δ cut consensus	Pipelined cut slots	$>$ 5 Ggas/s E*	$<$ 250 ms ordering*
Hermes (v2, Autobahn lanes) [85]	$O(n^2\hat{m} + \lambda n^3)$ bits/view	2δ ready view to prefix*	Prefix view with independent lanes	NR	NR
Multimmit (v4) [59]	$O(n)$ payload copies*	2δ proposal to order; mean 3δ from data*	$\sim 2\delta$ view with producer batches	NR	NR

Gatling preserves its component's confirmation path. Average latency from receipt at all nodes is $3\Delta + \Delta/K$. The 215 ms observation is from a run without failures. Gatling also reports 244.1 ms mean with 1% proposal drops. Neither observation is latency from submission to execution [92]. Mir's rate uses 100 nodes and requests of 500 bytes. Mir explicitly does not execute requests. ISS's displayed rate uses 32 nodes and one straggler. Its measured client endpoint is $f + 1$ replies. Ladon's message bound in the normal case does not resolve Gatling's published safety objection. Orthrus tests execution and abort outcomes. Payment and contract paths differ. Raptr's mean 5δ includes dissemination and average proposal wait. The 755 ms figure measures delay from mempool receipt until ready for execution at the displayed 250k tx/s. The separately reported peak is 260k tx/s.

Table 15. Performance of DAG data planes and ordering protocols.

Protocol	**Comm.**	**Protocol bound** start to endpoint	**Cadence unit / interval**	**Source rate**	**Measured latency**
Narwhal–HotStuff [31]	$O(n)$ payload copies; HotStuff control	NR uniform client bound	DAG rounds / HotStuff views	$\sim$ 140k tx/s O*	$<$ 2 s, client to leader commit*
Tusk [31,93]	Narwhal with no extra ordering messages	Expected $4.5R_D^*$	Wave start every $2R_D$	$\sim$ 160k tx/s O	NR
Bullshark (PSync) [100]	Amortized $O(n)$/tx*	$2R_D$ good case	Leader every $2R_D$	125k tx/s O	$\sim$ 2 s, U*
DAG-Rider [50]	Amortized $O(n)$/tx*	Expected $O(1)$ waves*	$4R_D$/wave	NR	NR
Mysticeti-C [7]	Uncertified DAG gossip	3δ, P*	DAG round gated by a quorum	$>$ 200k tx/s O	$\sim$ 0.5 s commit*
Fino [65]	$O(n^2)$ RB messages*	$2R_D$ for order and R_D for opening	Votes or complaints with retrieval	NR	NR

Narwhal–HotStuff's rate and latency are the one-worker measurements in Section 7.1 and Figure 6: 512-byte transactions, up to 500 KB batches, 10–50 validators across five AWS regions, without faults. Latency is client submission to commitment at the proposing leader; execution is not measured. The paper's separate four-RTT average-case model is not a worst-case client bound. Tusk's $4.5R_D$ assumes random message delays. The liveness failure in the specification identified by Shoup prevents interpreting it as a guarantee. Bullshark's measured endpoint is client submission to one party's commitment. Its certified rounds must not be replaced by single network delays. DAG-Rider's wave count concerns expected commit progress, not a fixed transaction's submission time. Both amortized costs require batching. DAG-Rider additionally uses coded dispersal (Bracha broadcast gives $O(n^2)$ instead). Mysticeti's three delays require data readiness and decidable preceding slots. Its benchmark uses arbitrary transactions of 512 bytes. Fino's costs concern reliable broadcast, amortized across batches. Its evaluation measures dispersal/retrieval overhead rather than full execution. None of these displayed O rates establishes executed goodput.

Table 16. Performance of permissionless probabilistic configurations.

Protocol	**Comm.**	**Protocol bound** start to endpoint	**Cadence unit / interval**	**Source rate**	**Measured latency**
SPECTRE [97]	Block/DAG gossip, byte total NR	N/A general total order	Mean PoW interval $1/\lambda_b$	NR	NR
GHOSTDAG [98]	Block/tip gossip, byte total NR	Depends on security depth	Mean PoW interval $1/\lambda_b$	NR	NR
Conflux [60]	Tree/DAG gossip, byte total NR	$d\,O(\log(1/\varepsilon))$	0.25 s mean block interval	3,480 tx/s E*	< 1 min*
Prism [9]	Separate data, voter, and proposer blocks	Expected $O(\log(1/\varepsilon))$ rounds	Multiple PoW block rates	$0.9C_h$ M*	NR
OHIE [116]	$O(\log k)$ hashes/block + payload	$\Theta(\Delta L)^*$	Mean $1/(pn)$ ticks/chain	4 to 10 Mbps O*	NR
Filecoin EC [81]	Block/message gossip, byte total NR	Depends on confidence	30 s epoch with possible nulls	NR	NR

Here a is adversarial hash share, d a propagation bound, ε confirmation error, and λ_b a mining rate. OHIE uses k chains, success probability p for each query, n miners, and $L = \log(1/\varepsilon) + \log k$. These are its model's query ticks. Conflux reports 9.6 Mbps of throughput from consensus alone using 20 Mbps/node links, disables signature verification and execution and observes roughly 50 s confirmation. Its separate mixed workload includes execution and reaches 3,480 tx/s with confirmation under one minute. These results must not be combined into one operating point. Prism uses $C_h = (1-a)C$ for modeled honest network capacity. Its faster bound for honest payments is not a bound on a general total order. OHIE's prototype reports transaction bytes, not successful executed transactions. Filecoin epochs are not finality. The EC specification's illustrative confidence cutoff of 900 epochs and F3 finality are separate mechanisms.

E.6 Additional mechanisms and their requirements

Table 17. Additional mechanisms and their requirements. The third column identifies what each mechanism needs to provide a complete transaction service.

Mechanism	Direct effect	Additional obligation
Inclusion lists [106,105]	Constrain admissible omission	Timely propagation, validity, and capacity
Availability certificates [31,42]	Evidence for body recovery	Retention and timely retrieval
Delayed reveal [13,24]	Restrict access to contents	Binding before reveal and policy for missing shares
Coded submission [86]	Reduce redundant payload bytes	Decode threshold and collectible byte fees
Fair ordering [52,19,51]	Constrain order of selected inputs	Inclusion, cycle handling, and recovery
Builder delegation [77,30]	Separate construction from proposal	Independent paths and fair exchange
Fee allocation [40,91]	Reward specified services	Coalition and duplicate incentives (Section 7)
Admission budgets	Bound unsettled obligations	Routing flexibility and capital efficiency (Section 7.2)

F Admission Solvency and Capital Efficiency

Coded dissemination reduces replicated bytes, but admission must ensure that accepted work is paid for or backed by an enforceable reservation if the design promises to avoid unpaid service. Sedna requires payment by byte at admission. A validator admits a bundle only after its local accounting predicate and payment of the byte fee pass, and the paper describes the fee in H_{pub} as prepaid. Deduplication and decode failure do not reverse the fee for bundles that enter finalized history [86]. Sedna does not specify a deferred charge upon finalization or implement a prepayment ledger shared across lanes. The remaining systems question is how to realize its admission abstraction when several lanes act on state that is not yet globally settled. Several lanes may admit work against the same stale balance before any reservation or prepayment becomes globally visible. Charging only the eventual transaction can leave redundant or invalid work unpaid. Charging each bundle requires enforceable authorization and sufficient funds for all simultaneously outstanding obligations. User u has an admissible lane set S_u and a monetary cap $Q_{u,i}$ on unsettled charges at lane i. If admission enforces these caps, each charge is authorized and redeemable at most once, reserved funds cannot be spent elsewhere, and outstanding charges may become payable together, the conservative condition

$$R_u \geq \sum_{i \in S_u} Q_{u,i}$$

reserves enough balance for those obligations. This accounting condition is not a Sedna theorem. Its usefulness depends on bounding admission, retries, settlement delay, and cancellation. Limiting lane count alone leaves exposure unbounded if each lane can admit unlimited unsettled work. Coupling ordering with execution can shorten exposure without itself proving a capital bound. Subscriptions to bounded lane subsets, separately allocated budgets, or prepaid tickets could make obligations locally checkable. These options trade routing flexibility and censorship insurance against locked capital and coordination. A subscription tied to one producer also raises questions about payment channel redemption, disputes, withdrawal, and whether the atomic payable unit is a transaction, bundle, or shard. Pricing by byte alone does not specify these mechanisms. Repeated interaction offers another research direction. Throttling, reputation, or terminating unpaid service may reduce collateral needs, but requires analysis of identity replacement, false accusations, and censored users' ability to switch providers. Evaluation should report useful executed work under duplicate and insolvency attacks, unrecovered service cost, and capital locked per user. Peak throughput alone cannot establish useful service under unpaid attack traffic.

G Glossary

Term	Meaning
Validator	Participant that checks protocol messages and takes part in agreement; commonly also maintains and executes the ledger state.
Proposer	Participant authorized to form a candidate contribution carrying transactions.
Leader	Participant coordinating an agreement instance; it need not be the only proposer.
Builder	Party constructing a proposal or payload on behalf of a proposer.
Canonical	Accepted as part of the common ledger under the protocol's agreement and finality rules.
Reconciliation	The rules through which validators agree which proposed contributions form a ledger update.
Ledger step	One protocol-defined reconciliation event fixing new canonical contributions for an update or log segment.
Proposal	Authenticated record identifying an author, body reference, and protocol metadata.
Proposal body	Transactions and additional data needed to validate or execute the contribution.
Proposal reference	Binding identifier of a body; receiving it does not imply possession of the body.
Admissible transaction	Transaction and submission satisfying a service claim's timing, fee, capacity, and state premises.
Qualified proposal	Proposal meeting the protocol's selection requirements and the checks applicable at that stage.
Proposal admission	A proposer places a transaction in an authenticated candidate.
Canonical contribution membership	A proposal body or reference becomes a required contribution to a ledger step.
Canonical transaction inclusion	A transaction enters the canonical execution sequence under the protocol's recovery, ordering, and filtering rules.
Availability certificate	Authenticated attestations to a specified condition supporting recovery of a body or enough coded fragments.
Missing proposal	An expected path has supplied no acceptable proposal record by its cutoff.
Unavailable body	A selected reference exists, but a particular validator does not yet have the matching data.
Decision-bound history	Proposal records that the agreed decision and its interpretation rules permit to affect selection or ordering.
Partial body map	A validator's lookup table from selected references to bodies already recovered and verified.

Term	Meaning (continued)
Domain of the body map	Selected references with a verified body already entered in that table.
Unresolved set	Selected references whose bodies are still missing at the stated validator and time.
Consensus output	The agreed value interpreted to determine canonical contribution membership.
Lane and cut	A lane is a continuing proposal history; a cut specifies how far each lane contributes.
Causal proposal DAG	A directed acyclic graph whose proposal references determine canonical membership or an ordering not reducible to merging independently selected stream prefixes.
Transaction identity	Protocol-defined identifier used for duplicate and replay rules; it need not identify economic intent.
Deduplication	Handling repeated copies within the selected input according to a deterministic rule.
Replay filtering	Applying the protocol's rules to transactions already processed in earlier ledger history.
Binding point	Event after which an actor can no longer change the specified choice.
Bounded inclusion	The protected transaction or proposal reaches a named endpoint within a stated horizon under its receipt and admissibility premise.
Include-or-empty	The protected input is included whenever the relevant output is nonempty; empty runs need a separate progress bound.
Ordering finality	The canonical order becomes irreversible under the stated model and confirmation confidence.
State finality	The resulting state transition or authenticated state commitment becomes final under the protocol's rules.
Executed goodput	Distinct useful transactions or state transitions completed per unit time, under the stated workload and failure accounting.
Cadence	Interval between successive starts of a named production or decision unit.
Last-look protection	A restriction on using specified later information to change a protected choice after another participant's corresponding choice has closed.

H Pseudocode for the MCP Reconciliation Families

We give representative protocols for the four direct MCP families and for constrained assembly. Cryptographic primitives and established agreement protocols are invoked through their stated interfaces. The direct BFT examples use $n = 3f + 1$ validators, at most f static Byzantine faults, and quorum $q = 2f + 1$.

Constrained assembly inherits its base protocol's fault model. Quorums count distinct authenticated validators. Messages and proofs bind the chain, epoch, step, author, and message type. Invalid messages are ignored. One-shot actions execute once when their guards become enabled, including on late messages; recurring handlers run for each eligible object. Independent instances may overlap.

MVBA, CutOrder, and AnchorOrder include their native validity, locking, view-change, and recovery rules. Their safety and termination properties are assumptions of these compositions. The Cadence and illustrated stream/DAG compositions use partial synchrony; ACS uses asynchronous randomized agreement with its required shared randomness. Data holders retain and serve bodies under their availability contracts. Missing selected data remain pending; a local timeout never removes them from the canonical input.

Algorithms 1–5 prepare inputs for Algorithm 7. Constrained assembly uses its base protocol's native ledger rules. Deterministic merge choices are stated explicitly.

Execution inputs. Each family supplies the interpretation record I_s of Appendix B when it invokes $\mathsf{Deliver}(s, X_s, I_s)$. The record carries the agreed decision and verified contribution metadata needed to derive the native execution context, preserving the origin of each transaction occurrence.

H.1 Fixed proposal vectors

Instance. One Chorus slot s, with ordered proposer set P_s and deadline D. Conductor supplies the slot schedule, participation, and bounds on outstanding slots. Static faults, synchronized clocks, and post-stabilization delay bound Δ are assumed. Message-sending handlers run only while participating; otherwise their triggers remain pending. Receive-only processing may run earlier.

Evidence. A FastQC contains q matching first-round entry signatures. A $\mathsf{FallbackQC}$ contains $f + 1$ matching fallback-entry signatures. An $\mathsf{EquivCert}$ contains two conflicting roots signed by the same proposer for this slot. A positive entry is a root binding an encrypted batch. Empty entries are $\perp$.

Continue during fallback. A fallback pledge prevents sending a fast commit; it does not prevent relaying fast evidence or accepting a valid fast commitment certificate. Slot-key shares are released at D, before agreement. Optional speculative output is omitted.

MVBA validity. One instance MVBA_s accepts a complete vector with either (i) all FastQC entries, or (ii) one FastQC, $\mathsf{FallbackQC}$, or $\mathsf{EquivCert}$ per position and a certificate of q fallback pledges. This is an objective evidence check. The local preference order below is used to construct a proposal, not to reject another valid proposal.

Availability is part of the fallback. An MVBA decision alone does not authorize delivery. The additional chunk dissemination and fallback commitment round establish the needed recovery path. Empty entries require no chunk. A slow or silent proposer is resolved by evidence and agreement, never by a validator privately deleting its entry.

Algorithm 1 Fixed proposal vectors: Cadence proposal and fast path

For this slot: $path \leftarrow$ `NONE`; $mvbaStarted \leftarrow$ `false`.
For each $j \in P_s$: $first[j] \leftarrow$ `UNSET`; retain verified evidence.
At $D - \Delta$, at each designated proposer j:
    Sign my proposal; encrypt it under the threshold identity for slot s.
    Encode the ciphertext into n indexed chunks; any $f + 1$ reconstruct it.
    Commit to the chunks; sign the root, slot, and proposer identity.
    Send each validator its assigned chunk, proof, and signed root.
Upon receiving a verified chunk:
    Store it and enable background recovery.
    If it is my assigned chunk, arrived by D, and $first[j] =$ `UNSET`:
        $first[j] \leftarrow$ its root.
At or after D, once participating, exactly once:
    Replace every remaining `UNSET` entry in $first$ by $\perp$.
    Broadcast my signed first-round entries, their assigned chunks,
        and my verifiable *slot-key share*.
Upon receiving a first-round vector:
    Require a complete signed vector, valid assigned chunks for positive entries,
        and a valid slot-key share; retain the verified chunks and share.
    For each position with q matching entry votes, retain its FastQC.
Upon first knowing a vector B with a FastQC at every position:
    Adopt its FastQCs; **broadcast** B and its evidence, even after pledging fallback.
    If $path =$ `NONE`, atomically set $path \leftarrow$ `FAST` and
        broadcast `FastCommit`$(s, \mathsf{Entries}(B))$.
Upon q matching FastCommit votes, or their valid certificate:
    Relay the certificate; accept the vector once; call ResolveCadence (Algorithm 7).

Algorithm 2 Fixed proposal vectors: Cadence fallback

When time $\geq D + \Delta$, q valid first-round vectors are known, and $path = \texttt{NONE}$:
- Atomically set $path \leftarrow \texttt{FALLBACK}$; this permanently forbids `FastCommit`.
- For each designated proposer j:
  - If I know a FastQC for its entry, include that certificate.
  - Else if $f + 1$ positive first-round votes name one root r and its ciphertext decodes and re-encodes to exactly r:
    - Send every validator its assigned chunk and proof.
    - Include my signed fallback vote for r and the proposer's signature on r.
  - Else include my signed fallback vote for $\perp$.
- **Broadcast** these entries and my signed no-fast-commit pledge.

Upon fallback messages:
- Retain each sender's first valid complete message; harvest attached FastQCs.

Upon q valid fallback messages, if $mvbaStarted = \texttt{false}$:
- For each position, choose the first available evidence: a FastQC; otherwise an EquivCert; otherwise $f + 1$ matching fallback votes.
- Attach the q-pledge certificate unless every entry has a FastQC.
- $mvbaStarted \leftarrow \texttt{true}$; propose the resulting vector to MVBA_s.

When time $\geq D + 2\Delta$, a fully FastQC-certified vector B is known, and $mvbaStarted = \texttt{false}$:
- $mvbaStarted \leftarrow \texttt{true}$; propose B to the *same* MVBA_s.

Upon MVBA_s deciding a valid vector B:
- For every positive entry backed by a FallbackQC:
  - Wait for my assigned verified chunk; **rebroadcast** that chunk.
- **Broadcast** $\texttt{FallbackCommit}(s, \mathsf{Entries}(B))$.

Upon q matching FallbackCommit votes, or their valid certificate:
- Relay the certificate; accept the vector once; call ResolveCadence (Algorithm 7).
- Interpret equivocation entries as empty in the recovered proposal vector.

H.2 Selected proposal collections

Primitives. Each epoch s has one reliable broadcast $\mathsf{RBC}_{s,j}$ and asynchronous binary agreement $\mathsf{ABA}_{s,j}$ per validator j. RBC supplies integrity, agreement, correct-sender validity, and totality. ABA supplies agreement, validity, and probabilistic termination with its required coin setup. Process ABA messages even before supplying local input; supply that input at most once.

Algorithm 3 Selected proposal collections: HoneyBadger-style ACS

At the start of epoch s, at every validator i:
For all j: $cipher[j] \leftarrow$ `UNSET`; $bit[j] \leftarrow$ `UNSET`; $inputSent[j] \leftarrow$ `false`.
Choose a batch using the configured pending-transaction sampling policy.
Start $\mathsf{RBC}_{s,i}$ with its threshold-encrypted ciphertext.
Participate in every $\mathsf{RBC}_{s,j}$ and $\mathsf{ABA}_{s,j}$.
Upon $\mathsf{RBC}_{s,j}$ delivering ciphertext c:
$cipher[j] \leftarrow c$.
If $inputSent[j] =$ `false`: set it true and input 1 to $\mathsf{ABA}_{s,j}$.
Upon $\mathsf{ABA}_{s,j}$ deciding b:
$bit[j] \leftarrow b$.
If at least $n - f$ entries of bit equal 1:
    For every h with $inputSent[h] =$ `false`:
        Set $inputSent[h] \leftarrow$ `true` and input 0 to $\mathsf{ABA}_{s,h}$.
Upon all n ABA instances deciding:
$S \leftarrow \{j : bit[j] = 1\}$; wait for RBC delivery of every $j \in S$.
Finalize the common collection $\{(j, cipher[j]) : j \in S\}$.
Only after collection selection, for every $j \in S$ in parallel:
If $cipher[j]$ is publicly malformed, set $batch[j] \leftarrow \perp$.
Otherwise broadcast my decryption share bound to $(s, j, cipher[j])$.
    Collect $f + 1$ distinct valid shares; robustly decrypt $cipher[j]$.
    Set $batch[j]$ to its decoded batch, or $\perp$ for a deterministic malformed result.
When every selected ciphertext has resolved:
Let X_s be all transactions in the resolved batches, sorted by
    (canonical ID, proposer index, position within the batch).
Queue X_s for $\mathsf{Deliver}(s, X_s, I_s)$ (Algorithm 7), in epoch order.
After delivery, remove processed IDs from the pending queue; start epoch $s + 1$.
Continue protocol service needed for other correct validators to finish.

Required details. The zero-input trigger is $n - f$ *ABA decisions of one*, not $n - f$ RBC deliveries. Retain all selected instances; do not truncate S to the first $n - f$. Robust decryption must give a common plaintext or malformed result, including for adversarial ciphertexts. An omitted candidate is allowed; a selected one cannot be skipped while waiting for its body.

Ordering choice. Canonical transaction-ID sorting, with ties broken by proposer index and then position within the batch, is the explicit deterministic merge used here. The source's backlog analysis additionally depends on its

sampling policy: sample $\lfloor b/n \rfloor$ transactions from the oldest b pending transactions. A large common subset alone does not protect a particular proposer.

H.3 Persistent proposal streams

State and contract. Each lane has a common genesis. Set $last[i] = 0$, the last processed position in lane i. A possession certificate PoA has $f + 1$ votes and proves recoverability under static faults and retention. It does *not* prove non-equivocation. CutOrder invokes Autobahn's complete consensus engine, including its prepare/confirm phases, fast path, and view changes. This presentation selects certified tips only.

Byzantine lane boundary. A recovered suffix can follow a different fork from the previously delivered hash; ancestors at positions $\leq last[i]$ are ignored. Raw cuts from parallel slots can regress, while *last* is monotone. The default $n - f$ new-tip coverage policy is a leader policy, not a cut-validity rule. Genesis is treated as already acknowledged.

H.4 Causal proposal DAGs

Instance. A Narwhal-style certified graph composed with an existing AnchorOrder protocol, such as HotStuff. A vertex binds (epoch, author, round, parent certificates, batch). Its certificate has q availability signatures. Initialize round $r = 1$, common genesis certificates at round 0, $voted[j, r] = \texttt{NONE}$, and $done = \varnothing$. Retain the graph; garbage collection is outside this simple presentation.

What is selected. The canonical set is the committed anchor's closure minus prior output. An unrelated locally visible vertex is not added. Missing ancestors stay pending; a local timeout cannot prune the closure. A certificate is availability evidence, not an anchor-finality proof.

Family boundary. The tie-breaker is a concrete deterministic traversal choice. Other DAG protocols need their own vertex, voting, and commit rules; this algorithm does not stand in for Mysticeti, DAG-Rider, or Bullshark. Inclusion requires suitable reference and anchor-selection policies.

H.5 Constrained assembly

Instance. Slot-s lists constrain the single block of slot $s + 1$. Use the configured committee, list limits, attester freeze T_f, builder freeze $T_b > T_f$, and next-slot attestation deadline T_a. Initialize local $admitted_s$ and $equiv_s$ sets empty. Timing assumes enough propagation slack for admitted lists to reach the builder and relevant evidence to reach attesters. Base supplies block consensus, fork choice, and finality with the inclusion check integrated.

Predicate contract. EnforceableAndAppendable is the versioned execution-layer omission check, including remaining gas and the specified post-payload validity tests. Test every missing transaction against the *same* (Σ_B, g_B); do not update that state while checking omissions. The builder loop is an illustrative assembly policy, not a prescribed ordering rule.

Algorithm 4 Persistent proposal streams: Autobahn-style lanes

When my preceding lane proposal is certified and a batch is ready:
$p \leftarrow$ signed (lane, next position, batch, parent hash, parent PoA).
Broadcast p; lane production does not wait for global agreement.
Upon a valid lane proposal p**:**
If its parent is missing: buffer p, recover the parent, and retry.
If I voted for its exact parent and have not voted at this lane/position:
    Store p and its required history; record the parent PoA.
    Send the owner my possession vote for (lane, position, hash of p).
Upon $f + 1$ **matching possession votes for my proposal:**
Form its PoA; publish it or carry it in the next proposal; enable the next position.
When authorized to propose cut C **for slot** s**, under the coverage policy:**
For each lane i, set $C[i]$ to my highest known certified tip.
Use its old/genesis tip if it has not advanced; submit C to CutOrder.
While processing a cut proposal under CutOrder:
Verify tip identities and certificates; start missing-history recovery in parallel.
Follow the engine's safe-voting rules; body recovery is outside its critical path.
Upon CutOrder deciding cut C **for slot** s**:**
Record the decision; process cuts sequentially in slot order through the *last* update.
For every lane i in parallel:
    If $C[i].position \leq last[i]$: $suffix[i] \leftarrow [\,]$.
    Else request positions $last[i]+1, \ldots, C[i].position$ in the chosen tip's ancestry
        from its PoA signers; retry until the authenticated range is recovered.
        Verify signatures, positions, hashes, and linkage within the range to the tip.
        Set $suffix[i]$ to that range in increasing lane position.
Once every required suffix has been recovered:
Initialize $X_s \leftarrow [\,]$; repeatedly visit lanes in fixed order.
On each visit, remove the next batch from that lane's nonempty suffix and append
    its transactions to X_s, preserving their order.
After all suffixes are exhausted, await Deliver(s, X_s, I_s) (Algorithm 7).
Set $last[i] \leftarrow \max(last[i], C[i].position)$ for each lane; process the next decided slot.
Upon a retrieval request: serve retained authenticated data and its history.

Algorithm 5 Causal proposal DAGs: certified vertices and ordered anchors

Upon entering round r with certificates from q distinct authors at $r-1$:
If I have not proposed at r, form a vertex v containing those parent certificates and my batch; sign and **broadcast** `Vertex`(v).

Upon receiving a signed vertex v from author j:
Verify its author, epoch, format, and binding to its batch and parent references.
Require valid parent certificates from at least q distinct authors at $v.round - 1$.
Process all parent certificates through **ADVANCE** before the next test.
If $v.round = r$ and $voted[j, r] =$ `NONE`:
Store the vertex durably; set $voted[j, r] \leftarrow hash(v)$.
Send j my signed `Available`(epoch, j, r, $hash(v)$).

Upon q matching Available signatures for my vertex v:
Form its availability certificate C and **broadcast** C.

Upon learning a valid certificate C:
Store it; request its vertex if absent.
If any round $u \geq r$ has certificates from q distinct authors:
Set r to one plus the greatest such u; enable proposal for that round.

When AnchorOrder requests an eligible proposal:
Submit a certified vertex reference under its proposal policy.
Perform all votes, locking, timeouts, and view changes under AnchorOrder's rules.

Upon AnchorOrder finalizing anchor C at log position s:
Process anchors sequentially in log order, through the update to *done*.
Fetch and verify the anchor and its complete referenced ancestor closure.
$U \leftarrow \mathsf{Closure}(C) \setminus done$, excluding empty genesis vertices.
Topologically order U with parents before children, breaking ties by the fixed key (round, author, vertex hash).
Concatenate the batches in that order, preserving each batch's internal order, into X_s.
Await $\mathsf{Deliver}(s, X_s, I_s)$ (Algorithm 7); set $done \leftarrow done \cup U$.

Upon a request for retained data: return the vertex and its referenced certificates.

Algorithm 6 Constrained assembly: FOCIL-style inclusion lists

During slot s, at each selected list author:

Choose a bounded list L; sign and gossip one $\texttt{List}(s, committeeRoot, author, L)$.

Upon a list received before T_a:

Verify slot, committee, author signature, and structural/list-size rules.
Record and relay under the duplicate and two-distinct-list gossip limits.
Builders separately cache valid lists until T_b; freeze their list snapshot then.
If two distinct authenticated lists exist for this author and slot: add it to $equiv_s$.
Else, if arrival is before T_f: add L to $admitted_s$.
After T_f, admit no new obligations; equivocation evidence may still remove an author.

At the builder's cutoff T_b:

$\Gamma_b \leftarrow$ its collected eligible lists excluding known equivocators.
Construct a base-valid candidate B extending the selected parent.
While a transaction in Γ_b is absent and appendable under the configured check:
    Append one under the builder's ordering policy; update state and remaining capacity.
Provide B to the slot-$(s+1)$ proposer.

At slot $s+1$: the proposer publishes B under Base's normal proposal rules.

When Base requests my support decision for the slot-$(s+1)$ block B:

$\Gamma_i \leftarrow admitted_s$ excluding authors in $equiv_s$.
Obtain B; perform Base's execution and data-availability checks.
Compute its post-payload state Σ_B and remaining capacity g_B.
$satisfied \leftarrow \texttt{true}$.
For each distinct transaction t in Γ_i:
    If t is absent from B and $\mathsf{EnforceableAndAppendable}(t, B, \Sigma_B, g_B)$:
        $satisfied \leftarrow \texttt{false}$.
Allow support for B only if $satisfied$ and Base otherwise permits it.
Follow Base's specified alternative fork-choice/attestation behavior otherwise.

When Base finalizes a block:

Recover and apply that canonical block under Base's native ledger rules.
Local list observations do not add transactions to the finalized body.

After T_a: stop accepting lists and equivocation evidence for source slot s.

Local views matter. Correct validators' Γ_i may differ. Failure of this check affects support under fork choice; it is not an extra objective execution-invalidity rule or a new quorum-based finality mechanism. Base supplies recovery and finality.

H.6 Recovery and ordered delivery

Upper-layer convention. The direct-family examples use a common first-occurrence duplicate rule and a deterministic state transition with the canonical context defined in Appendix B. Each call to Deliver supplies the sequence X_s and its interpretation record I_s, assembled from that protocol's agreed decision, canonical proposal metadata, and recovered bodies or agreed failure outcomes. This record preserves each occurrence's proposal origin where context depends on it. Substitute native replay or replacement rules consistently where different. FOCIL uses its base ledger semantics. Queuing an input is nonblocking; awaiting Deliver waits until that step is locally marked delivered.

Execution outcomes. Distinct signed transactions that spend the same balance or nonce are conflicts, not duplicate IDs. The deterministic transition returns success, rejection, or reversion according to the application. Inclusion does not promise successful execution. An unresolved earlier step holds back this delivery queue even if later decisions are already final.

Algorithm 7 Shared recovery, ordered delivery, and execution

In the background, for every positive Cadence root r:
Collect $f + 1$ chunks with distinct verified indices; decode ciphertext c.
If decoding fails, or re-encoding c does not reproduce the commitment r:
    Record a deterministic `INVALID` result.
Otherwise:
    Wait for $f + 1$ distinct verified slot-key shares and decrypt c.
    Check proposal encoding, slot, author, and signature; retain the proposal or `INVALID`.
ResolveCadence(s, B)**, after accepting a commitment certificate:**
Wait until every positive entry is recovered or deterministically `INVALID`.
Form vector V: use $\perp$ for empty/equivocation entries, invalid data, or author mismatch.
Finalize V once; notify Conductor; abandon this slot's MVBA instance.
Concatenate nonempty batches in fixed proposer order, preserving their internal order.
Queue the resulting X_s for $\mathsf{Deliver}(s, X_s, I_s)$.
Persistent state: initialize $next$, $\mathcal{L}$, Σ, and $Seen$ from genesis or the last checkpoint.
Initialize $ready$ empty or restore pending inputs consistently with that checkpoint.
Upon receiving a fully resolved input (X_s, I_s)**:** store it as $ready[s]$.
At the single delivery worker, while $ready[next]$ **exists:**
Let $(X, I) \leftarrow ready[next]$; $T \leftarrow [\,]$; $within \leftarrow \varnothing$.
For each transaction t in X, in its prescribed order:
    $id \leftarrow H(\text{chain domain, canonical signed encoding of } t)$.
    If $id \notin Seen \cup within$: append t to T; insert id into $within$.
Derive $C \leftarrow \mathsf{Context}_{next}(\mathcal{L}, \Sigma, I, T)$ from the canonical inputs.
Append T as the canonical transaction sequence for this step.
Set $(\Sigma, R) \leftarrow \mathsf{Execute}(\Sigma, T, C)$; retain the outcomes R.
Extend $\mathcal{L}$ with the canonical record for this step.
Update $Seen$ with processed IDs under this configured replay rule.
If separate state attestation is configured, submit the resulting state commitment
    to that protocol; report state finality only at its specified endpoint.
Mark the step delivered; delete its queued input; increment $next$.