DracoBFT Side Loops: Threshold-Attested External Inputs for BFT Financial Systems

Madhur Kumar Sharma[1⋆], Hiten Jain[1], and Vyom Sharma[1⋆⋆]

Hotstuff Labs
madhur@hotstufflabs.com, hiten@hotstufflabs.com, vyom@hotstufflabs.com

Abstract. Financial state machines consume facts that consensus does not create: market prices, deposits finalized on another chain, and balances held at external venues. Fetching those facts inside a block executor makes progress depend on variable-latency services; accepting a leader's signed answer instead preserves speed but makes that leader an oracle. We describe DracoBFT side loops, a prototype architecture in which the validator set runs service-specific observation and attestation protocols alongside a two-chain BFT substrate. A side loop produces an ordinary state-machine transaction only after collecting stake-weighted approval for a view-bound payload. We study three implemented instances: multi-source price aggregation, Ethereum deposit admission, and multi-venue equity snapshots.

The implementation study exposes an important distinction between a quorum that helped construct an input and evidence that every replica can verify when admitting it. Deposit and equity transactions retain aggregate signatures; the current price path retains only the leader's final signature. We formalize an evidence-complete envelope, prove conditional certificate uniqueness, at-most-once application, and a weighted-median validity bound, and give a domain-separated state-difference commitment. The results are design-level theorems under explicit assumptions, not a proof of the current Go prototype. We report the prototype-to-design gaps and a reproducible evaluation plan rather than unsupported throughput or production-readiness claims.

Keywords: Byzantine fault tolerance · external inputs · threshold attestations · blockchain oracles · financial infrastructure

1 Introduction

Byzantine state-machine replication establishes agreement on a deterministic sequence of commands. It does not establish the truth of an exchange price, an Ethereum event, or an account balance returned by a trading venue. A financial blockchain nevertheless needs these values to calculate margin, recognize deposits, settle funding, and account for assets held outside its local state. Calling remote

⋆ University of Waterloo alumnus, Computer Engineering.
⋆⋆ Corresponding author.

services during block execution violates determinism and can stall consensus. Trusting a single relayer avoids that failure mode but turns the relayer into a privileged oracle.

DracoBFT explores a middle design point. Its *side loops* are service-specific protocols run by the consensus validators outside the ordering loop. Validators independently obtain or check an observation, sign a view-bound digest, and deliver their votes to a side-loop leader. The leader turns a quorum result into a normal transaction. Consensus remains the only component that orders and applies state changes; a side loop can propose a transaction but cannot force a commit.

This pattern resembles oracle aggregation, but it is integrated with the validator set, view schedule, transaction verifier, and finalized-state callbacks. That integration creates subtle obligations. The side-loop vote threshold must use the same stake snapshot as transaction admission. The transaction must retain enough evidence for every replica to validate it. Observation policies must distinguish objective evidence, such as a finalized receipt, from approximate agreement, such as two venue balances sampled at nearby times. Side-loop callbacks also need resource isolation; otherwise an "asynchronous" service can still delay the consensus goroutine.

This paper makes four contributions:

1. It extracts a generic side-loop protocol from a running Go codebase and identifies the minimum evidence envelope required for deterministic admission.
2. It analyzes three concrete loops—price feeds, deposits, and external venue equity—and states the distinct trust and liveness assumptions of each.
3. It proves conditional quorum-intersection, certificate-uniqueness, at-most-once, and weighted-median properties, and states precisely which implementation obligations are outside those proofs.
4. It reports gaps found by mapping the model to the implementation and defines the tests and measurements needed for a complete systems evaluation.

DracoBFT does not introduce a new BFT commit rule or proof system. It uses a Fast-HotStuff-style substrate [6] and treats its safety and liveness as assumptions. zkTLS, succinct cross-chain proofs, provider markets, and regulated payment routing are compatible future adapters, not implemented results of this paper.

2 System and Threat Model

Let $\mathcal{V}_e$ be the validator set in epoch e, with positive stake weights totaling W_e. Byzantine validators have total weight $F_e < W_e/3$. A quorum has weight $Q_e \geq 2W_e/3$. Channels between validators are authenticated, signatures are unforgeable, and the network is partially synchronous [2]. Reconfiguration is performed by the inherited consensus substrate and activates at an unambiguous finalized boundary.

The substrate exports an ordered deterministic transaction log and a finalized block identifier. We assume substrate safety: honest replicas do not finalize conflicting blocks. We assume substrate liveness only after network stabilization and only when a valid transaction remains available to honest leaders. These assumptions must be discharged against the exact Fast-HotStuff implementation before claiming end-to-end consensus correctness; HotStuff's quorum intersection alone is insufficient to justify a modified vote or timeout rule [10,6,3].

A side-loop instance is identified by

$$I = (\mathsf{chainID}, e, \ell, v, \rho, \mathsf{anchor}), \tag{1}$$

where ℓ is the loop type, v is its view or sequence number, ρ is a policy version, and anchor is a finalized block or an allowed anchor range. A payload x contains a canonical observation or aggregate result. The policy $\Pi_{\ell,\rho}$ defines eligible signers, a local validation relation $R_{\ell,\rho}(x, o_i)$ over validator i's observation o_i, freshness and size limits, aggregation semantics, and expiry behavior.

The adversary may control all clients, side-loop leaders, external messages, and validators of weight F_e. It can equivocate, replay old votes, omit observations, reorder messages, and return malformed data. It may also control an external origin. Consequently, a valid HTTPS response authenticates an origin's statement but not the statement's truth. An Ethereum receipt obtained through an RPC endpoint inherits that endpoint's chain view and the source chain's reorganization risk. Venue APIs may expose non-atomic snapshots.

We target four properties. *Evidence completeness* means transaction admission verifies the same statement approved by the side-loop quorum. *Instance uniqueness* means honest stake never certifies conflicting payloads for one instance. *At-most-once effect* means a certified instance changes application state at most once. *Consensus isolation* means external calls and unbounded proof work cannot execute in the consensus event loop. Ground-truth accuracy and availability of external services require adapter-specific assumptions and are not consequences of BFT agreement.

3 Evidence-Complete Side Loops

3.1 Generic protocol

An observation trigger is derived from a finalized anchor or deterministic block callback. Validator i obtains o_i, constructs or receives a candidate x, and evaluates $R_{\ell,\rho}(x, o_i)$. If it accepts and has not signed a different payload for I, it signs

$$d = \mathsf{H}(\texttt{DRACO-SL-V1} \| \mathsf{Encode}(I) \| \mathsf{H}(\mathsf{Encode}(x))). \tag{2}$$

The encoding is injective and length-prefixed. The tag, chain identifier, loop identifier, epoch, policy version, and anchor prevent cross-protocol, cross-chain, stale-policy, and replay reinterpretation. The leader collects a signature map Σ whose distinct signers have weight at least Q_e under the epoch-e stake snapshot.

It then submits $T = (I, x, \Sigma)$. Deterministic admission recomputes Eq. 2, checks every signature and signer once, reconstructs the applicable stake snapshot, checks $\mathsf{wt}(\Sigma) \geq Q_e$, enforces freshness and policy activation, and requires the on-chain instance state to be pending. Successful execution atomically records the result and marks I consumed. Large auxiliary evidence may be referenced through a commitment only if its availability is guaranteed before validators vote for the containing block.

The evidence must travel with, or be reproducibly derivable from, the admitted transaction. A leader signature stating that a quorum was previously observed is not a quorum certificate. This distinction is the principal lesson from the prototype mapping in Section 5.

3.2 Price aggregation

Each validator samples a configured set of exchanges and on-chain feeds, normalizes symbols and units, and reports (I, p_i, t_i, S_i), where p_i is a price vector, t_i is a bounded timestamp, and S_i records source coverage. The certificate should preserve the signed reports or a binding aggregate with individually verifiable membership. For each instrument, the deterministic result is a stake-weighted median of reports satisfying freshness, range, and minimum-source-diversity rules.

Median aggregation limits extreme values only under an observation assumption: honest reports for an instrument must lie in an acceptable interval. It does not protect against a market-wide bad source, correlated exchange failures, or honest validators sampling rapidly moving markets at incompatible times. Source-level medians inside each validator and validator-level weighted medians answer different adversaries and should both be reported.

3.3 Cross-chain deposits

The deposit adapter observes a source-chain event containing source chain, contract, transaction hash, block hash and height, log index, token, amount, receiver, and application flags. Honest validators sign only after applying one fixed source-chain finality rule and verifying event inclusion. An RPC receipt lookup is useful operationally but is not a self-contained proof: replicas may share a faulty provider or observe different canonical heads. A hardened adapter therefore includes either a light-client-verifiable header and receipt proof, or explicitly labels the result a threshold attestation to RPC-observed state. The destination chain stores a unique source-event key and rejects a second application regardless of batch or destination transaction identifiers.

3.4 External-venue equity

The equity adapter samples balances for configured venue accounts and accepts a leader snapshot when local observations are within a policy tolerance. Unlike a receipt, an account balance changes continuously and may not have a common

cryptographic anchor. The certificate therefore proves only that a stake quorum signed compatible observations for a named sampling window. It does not prove solvency, ownership of venue assets, or withdrawal availability. Canonical account ordering, decimal rules, timestamps, missing-account behavior, and staleness must be part of the signed policy statement. Threshold signatures provide accountability; periodic venue statements, reserve proofs, or TEEs can strengthen the underlying evidence but introduce separate assumptions.

4 Conditional Security Analysis

The following results concern the evidence-complete protocol above. They assume substrate safety, the epoch stake snapshot, injective encoding, collision resistance, unforgeable signatures, and honest validators following the sign-once rule. They are not presented as a refinement proof of the Go code.

Lemma 1 (Weighted quorum intersection). *For any two quorums $A, B \subseteq \mathcal{V}_e$ with weight at least Q_e, $\mathsf{wt}(A \cap B) \geq 2Q_e - W_e \geq W_e/3 > F_e$.*

Proof. Because $\mathsf{wt}(A \cup B) \leq W_e$, $\mathsf{wt}(A \cap B) = \mathsf{wt}(A) + \mathsf{wt}(B) - \mathsf{wt}(A \cup B) \geq 2Q_e - W_e$. The remaining inequalities follow from $Q_e \geq 2W_e/3$ and $F_e < W_e/3$. Hence every two quorums intersect in honest weight.

Theorem 1 (Certificate uniqueness). *Two conflicting payloads cannot both obtain valid certificates for the same instance I.*

Proof. Suppose payloads $x \neq x'$ have certificates from signer sets A and B. The intersection lemma gives honest weight in $A \cap B$. Since Eq. 2 binds I and the payload hash, every validator in the intersection signed both conflicting digests. This contradicts the honest sign-once rule, except with a hash collision, encoding ambiguity, or signature forgery.

Theorem 2 (At-most-once application). *A side-loop instance causes at most one successful application transition.*

Proof. Admission requires the replicated instance state to be pending. The successful transition atomically records $\mathsf{consumed}(I)$ with the application effect. Substrate safety gives honest replicas one transaction order. Every later replay observes the consumed state and fails before producing an effect. Certificate uniqueness additionally prevents two conflicting certified values from competing for the first transition.

Lemma 2 (Weighted-median validity). *Let a certificate contain report weight $q \geq 2W_e/3$. If Byzantine report weight is less than $W_e/3$ and every honest reported value lies in $[L, U]$, then every stake-weighted median of the certified reports lies in $[L, U]$.*

Proof. Byzantine certified weight is less than $W_e/3 \leq q/2$. A median below L would require at least half the certified weight at or below that value, but only Byzantine reports can be below L. The symmetric argument rules out a median above U.

This lemma does not apply to an unweighted median under a stake-bounded adversary: many low-stake Byzantine identities can dominate the sample count. It also provides no bound when honest values do not share an interval. These conditions must be measured for the price adapter rather than assumed.

For deterministic evidence, such as a source event under a fixed light-client rule, Byzantine weight alone cannot certify an invalid payload because its weight is below the quorum threshold. For subjective or time-varying evidence, a certificate establishes quorum attestation, not external truth. The paper therefore avoids the term "trustless" for venue and RPC-backed results.

4.1 State-difference commitment

The prototype also chains per-block state-change hashes. A hardened definition collapses repeated writes to each canonical key, sorts keys by byte encoding, and computes

$$D_v = \mathsf{H}(\texttt{DRACO-DELTA-V1} \| \mathsf{Encode}(\Delta_v)), \tag{3}$$

$$C_v = \mathsf{H}(\texttt{DRACO-CHAIN-V1} \| \mathsf{chainID} \| v \| b_v \| C_{v-1} \| D_v), \tag{4}$$

where b_v is the finalized block identifier. If two different committed transcripts produce one C_v, then, working backward from the first differing tuple, either the canonical encoding is not injective or a collision in H has been found. This commitment binds ordered differences; it does not authenticate an initial snapshot or provide membership proofs. An authenticated checkpoint such as a Merkle commitment is still required for trustless state bootstrap [7,8].

5 Prototype Mapping and Required Evaluation

The Go prototype separates the Fast-HotStuff event loop from a global-event manager with price-feed, deposit, and multi-venue handlers. Finalized/proposed block callbacks advance side-loop views; handlers exchange proposals and votes; successful loops broadcast ordinary transactions. Table 1 records the result of mapping the generic protocol to the implementation. It is a gap analysis, not a production certification.

The accompanying Quint model covers the consensus and global-event abstraction, not the full evidence envelope. A source audit finds 189 declared scenario runs across nine test modules. All 14 source and test modules type-check with Quint 0.32.0. After repairing one missing property import and two fixtures that tried to add an existing validator, ten TypeScript-simulator samples per discovered run completed successfully (323 executions including imported duplicates).

Table 1. Prototype-to-design mapping. "Required" items are blockers for the corresponding end-to-end claim.

Component	Prototype behavior	Required for evidence-complete admission
Deposit	Validators check receipts and the transaction retains a stake-quorum signature.	Bind block hash/finality and all event fields; document RPC trust or include an inclusion proof.
Price feed	A stake quorum sends signed local feeds; the leader submits a median with only its signature.	Carry quorum evidence, use stake-weighted aggregation, and enforce freshness/source coverage deterministically.
Venue equity	Validators threshold-sign values accepted within a 0.5% local tolerance.	Bind sampling window and policy; canonicalize ordering; define missing/stale data and tolerance semantics.
Change log	Chains the previous commitment with the finalized change-log hash.	Add domains, chain/view/block binding, canonical delta vectors, test vectors, and non-panicking recovery.
Isolation	External work generally starts in goroutines, but consensus waits for registered callbacks to return.	Enforce bounded callbacks, queues, cancellation, budgets, and backpressure tests.

Random execution is useful for finding counterexamples but is not a proof over all traces; bounded `quint verify` configurations and a model-to-Go conformance map are still required [4,5]. The Go unit tests for the price-feed, deposit, and multi-venue packages also pass at the audited artifact snapshot.

A complete evaluation should compare the unmodified consensus path with each side loop enabled at 4, 7, 10, and 16 validators across local and multi-region networks. It should report transaction throughput; finality p50/p95/p99; side-loop completion and expiry; callback time; queue depth; CPU, bytes, and signature-verification cost. Fault experiments should include a crashed or equivocating side-loop leader, stale and divergent sources, missing instruments, source-chain reorganization, RPC disagreement, delayed votes, duplicate events, validator reconfiguration, restart, and state catch-up. Price experiments must compare unweighted and stake-weighted aggregation under stake-bounded Sybil distributions. Deposit experiments must retain block and receipt fixtures so every result is replayable.

Reproducibility requires pinned source commits, Go and Quint versions, deployment manifests, validator stakes, source configurations, raw observations, seeds, and plotting scripts. Until those measurements and the required fixes exist, the defensible result is an architecture and implementation study, not measured sub-second external settlement or production readiness.

6 Related Work

PBFT established practical Byzantine replication, while HotStuff introduced a leader-based chained protocol with linear communication and optimistic responsiveness [1,10]. Fast-HotStuff and Jolteon reduce optimistic commit latency through two-chain designs with different timeout and recovery rules [6,3]. Side loops do not alter these rules; they produce inputs for an inherited substrate.

Town Crier authenticated HTTPS data using trusted hardware, and DECO proved statements about TLS sessions without server cooperation [11,12]. These approaches strengthen provenance for an individual observation. Side-loop certificates instead govern which observation statement a BFT state machine admits; an adapter can use a TLS proof as its local evidence. zkBridge similarly shows how succinct proofs can verify cross-chain state [9]. Replacing RPC receipt checks with such proofs changes the adapter's trust model but not the evidence-complete admission requirement.

Oracle aggregation and side loops share the need to handle correlated sources, freshness, omission, and strategic reporting. The distinctive engineering issue here is coupling auxiliary attestations to consensus views and stake snapshots without letting auxiliary work or unverifiable leader summaries enter the consensus-critical path.

7 Conclusion

Side loops can keep variable-latency observations outside BFT ordering while retaining validator accountability, but only when the finalized transaction carries the statement and evidence that replicas actually verify. The prototype demonstrates three useful integration patterns and also shows why quorum-assisted construction is weaker than evidence-complete admission. The next milestone is to implement the hardened envelope, repair the model and test gate, and measure the resulting system under divergent sources and Byzantine leaders.

Competing Interests

The authors are affiliated with Hotstuff Labs, which develops the prototype described in this paper and may benefit from its adoption.

References

1. Castro, M., Liskov, B.: Practical byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation. pp. 173–186 (1999)
2. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. Journal of the ACM **35**(2), 288–323 (1988). https://doi.org/10.1145/42282.42283

3. Gelashvili, R., Kokoris-Kogias, L., Sonnino, A., Spiegelman, A., Xiang, Z.: Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback. arXiv preprint arXiv:2106.10362 (2021), https://arxiv.org/abs/2106.10362
4. Informal Systems: Quint: An executable specification language. Project documentation and source code (2026), https://quint-lang.org/
5. Informal Systems: What does quint do? model checker and simulator. Quint documentation (2026), https://quint.sh/docs/what-does-quint-do
6. Jalalzai, M.M., Niu, J., Feng, C., Gai, F.: Fast-hotstuff: A fast and resilient hotstuff protocol. arXiv preprint arXiv:2010.11454 (2020), https://arxiv.org/abs/2010.11454
7. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Advances in Cryptology—CRYPTO '87. pp. 369–378 (1988). https://doi.org/10.1007/3-540-48184-2_32
8. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum Yellow Paper (2014), https://ethereum.github.io/yellowpaper/paper.pdf
9. Xie, T., Zhang, J., Cheng, Z., Zhang, F., Zhang, Y., Jia, Y., Boneh, D., Song, D.: zkbridge: Trustless cross-chain bridges made practical. Cryptology ePrint Archive, Paper 2022/073 (2022), https://eprint.iacr.org/2022/073
10. 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
11. Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: Town crier: An authenticated data feed for smart contracts. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 270–282 (2016). https://doi.org/10.1145/2976749.2978326
12. Zhang, F., Maram, D., Malvai, H., Goldfeder, S., Juels, A.: DECO: Liberating web data using decentralized oracles for TLS. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1919–1938 (2020). https://doi.org/10.1145/3372297.3417239