PATENT APPLICATION

INVENTOR(S)

0001 KIMZEY, SAMUEL C

TITLE

0002 Multi-Perspective Distributed Computation via MAC-Seeded Cryptographic Hash Differentiation

CROSS-REFERENCE TO RELATED APPLICATIONS

0003 This application claims the benefit of priority under 35 U.S.C. § 119(e) to the following provisional patent applications:

a. U.S. Provisional Application No. 63/810,843, filed 05/23/2025, titled “BlockMesh Decentralized Memory-Mesh Architecture”
b. U.S. Provisional Application No. 63/810,834, filed 05/23/2025, titled “Memory Field Pressure & Tension System”
c. U.S. Provisional Application No. 63/810,841, filed 05/23/2025, titled “Non-Generative Artificial Cognition Engine”
d. U.S. Provisional Application No. 63/810,837, filed 05/23/2025, titled “Proof of Memory Flow System”
e. U.S. Provisional Application No. 63/811,761, filed 05/23/2025, titled “Composite Cognition Pools for Emergent Memory-Driven Systems”
f. U.S. Provisional Application No. 63/811,758, filed 05/25/2025, titled “Genesis Document Anchor Method for AI Behavior”
g. U.S. Provisional Application No. 63/811,763, filed 05/25/2025, titled “On-Chain VM for Deterministic Language Model Execution”
h. U.S. Provisional Application No. 63/812,634, filed 05/27/2025, titled “Stateless Action Engine & Minimal Proof Logging”

i. U.S. Provisional Application No. 63/812,620, filed 05/27/2025, titled “One Action-One Mint Transaction Paradigm”
j. U.S. Provisional Application No. 63/812,645, filed 05/27/2025, titled “Epoch Rhythm & Synchronization Mechanism”
k. U.S. Provisional Application No. 63/814,230, filed 05/29/2025, titled “Agent Lifecycle Dormancy Management”
l. U.S. Provisional Application No. 63/814,244, filed 05/29/2025, titled “Selective Memory Disclosure via Zero-Knowledge Proofs”
m. U.S. Provisional Application No. 63/814,257, filed 05/29/2025, titled “Identity Bound Memory Rights”
n. U.S. Provisional Application No. 63/816,926, filed 06/03/2025, titled “Memory Driven Autonomy for Physical Agents”
o. U.S. Provisional Application No. 63/816,924, filed 06/03/2025, titled “Post-Generative Decision Support Mesh”
p. U.S. Provisional Application No. 63/816,907, filed 06/03/2025, titled “Surplus Flow Redistribution System and Method”

All of the above applications are incorporated herein by reference in part or in their entirety.

TECHNICAL FIELD

0004 This invention relates to distributed computation, cryptographic hash differentiation, multi-perspective result collection, hardware-seeded computation diversity, and parallel distributed processing. The invention encompasses methods for obtaining diverse computational outputs from identical inputs by leveraging unique hardware identifiers, collecting and assembling multi-perspective results from network broadcasts, achieving cryptographic diversity

without randomness, and utilizing inherent node differentiation for parallel exploration of solution spaces.

BACKGROUND

0005 Distributed computation systems typically seek consensus - multiple nodes computing the same result to verify correctness. However, certain computational tasks benefit from diversity rather than consensus: exploring solution spaces, generating multiple perspectives, or producing varied outputs from identical inputs. Existing systems lack mechanisms for deterministically producing diverse outputs from identical inputs across distributed nodes.

0006 No existing system provides a unified framework for:

a. Producing diverse cryptographic outputs from identical inputs using hardware-specific seeds
b. Leveraging MAC addresses or equivalent unique identifiers as differentiation seeds
c. Collecting multiple unique perspectives from a single network broadcast
d. Assembling diverse outputs into composite computational results
e. Achieving cryptographic diversity without randomness or non-determinism
f. Utilizing node-specific hardware characteristics for computation differentiation
g. Parallel exploration of solution spaces through inherent node diversity
h. Deterministic reproduction of any node's output given knowledge of its seed

DETAILED DESCRIPTION

0007 This invention provides a comprehensive framework for distributed computation that produces diverse outputs from identical inputs by seeding cryptographic hash functions with node-specific hardware identifiers.

0008 MAC-Seeded Hash Differentiation Principle:

0009 The fundamental mechanism exploits the mathematical property that cryptographic hash functions produce entirely different outputs for even slightly different inputs. By prepending a unique hardware identifier (such as MAC address) to each computation payload, every node in a distributed network produces a unique output from identical application data:

a. Node_A_Output = keccak256(MAC_A || payload)
b. Node_B_Output = keccak256(MAC_B || payload)
c. Node_C_Output = keccak256(MAC_C || payload)

Where:

a. `MAC_X` is the unique hardware identifier of node X
b. `payload` is identical application data sent to all nodes
c. `||` denotes concatenation
d. Each output is entirely different despite identical payload

0010 Properties of MAC-Seeded Differentiation:

a. Determinism: Given the same MAC and payload, output is always identical - no randomness involved
b. Diversity: Different MACs produce statistically independent outputs - cryptographic guarantee
c. Unpredictability: Cannot predict one node's output from another's without knowing both MACs
d. Reproducibility: Any party with MAC and payload can reproduce exact output
e. Parallelism: All nodes compute simultaneously on identical payload
f. No Coordination: Nodes need not communicate; diversity is inherent

0011 Hardware Identifier Selection:

0012 The unique seed may be derived from various hardware identifiers:

a. MAC Address: Network interface hardware address, 48 bits, globally unique by design
b. CPU Serial: Processor serial number where available
c. Disk Serial: Storage device serial numbers
d. TPM Identity: Trusted Platform Module endorsement keys
e. Composite Seed: Hash of multiple hardware identifiers for enhanced uniqueness
f. Derived Seed: Hardware seed processed through key derivation for operational flexibility

0013 The selection criteria:

a. Uniqueness: High probability of global uniqueness across nodes

b. Stability: Identifier persists across reboots and sessions

c. Accessibility: Identifier readable without elevated privileges where possible

d. Privacy: Identifier can be hashed before use to prevent tracking

0014 Broadcast-Collect-Assemble Pattern

0015 The distributed computation follows a three-phase pattern:

Phase 1 - Broadcast:

a. Coordinator prepares computation payload containing application data

b. Payload is broadcast to network through peer-to-peer gossip

c. All receiving nodes obtain identical payload bytes

d. Broadcast mechanism may be transaction pool, message queue, or equivalent

Phase 2 - Collect:

a. Each receiving node computes `hash(local_MAC || payload)`

b. Resulting hashes propagate back through network

c. Coordinator subscribes to hash announcements

d. Unique hashes are accumulated with deduplication

e. Collection continues for configurable duration or until sufficient hashes received

Phase 3 - Assemble:

a. Collected hashes represent multi-perspective computation results
b. Assembly combines hashes based on application requirements
c. Combination methods include: selection, XOR, concatenation, voting, or custom logic
d. Assembled result incorporates contributions from multiple nodes
e. Result quality scales with number of unique contributors

0016 Dispatcher Architecture

0017 The system implements a dispatcher component managing the broadcast-collect-assemble cycle:

a. State Management: Tracks whether collection is active, prevents concurrent dispatches
b. Hash Deduplication: Maintains set of seen hashes to avoid counting duplicates
c. Result Accumulation: Stores unique hashes in both set and list forms
d. Timeout Handling: Enforces maximum collection duration
e. Early Termination: Supports stopping collection when sufficient results obtained
f. Callback Integration: Invokes application-provided hash processors for real-time filtering

0018 Collection Strategies

0019 Different applications require different collection approaches:

a. Time-Bounded: Collect for fixed duration, return all unique hashes
b. Count-Bounded: Collect until N unique hashes received
c. Quality-Bounded: Collect until hash meeting criteria found (e.g., leading zeros)
d. Hybrid: Combination of time, count, and quality bounds
e. Streaming: Process hashes as they arrive, no accumulation
f. Sampling: Probabilistically retain subset of received hashes

0020 Application: Semantic Coordinate Derivation

0021 In cognitive systems, MAC-seeded hashes provide multi-perspective semantic coordinates:

a. Single payload broadcast represents semantic query
b. Each node's hash maps to different coordinates in semantic space
c. Multiple coordinates provide diverse perspectives on query
d. Assembly selects or combines coordinates based on coherence metrics
e. Result reflects distributed consensus on semantic interpretation

0022 Application: Solution Space Exploration

0023 For search problems, MAC-seeded hashes provide parallel exploration:

a. Payload encodes problem parameters
b. Each node's hash represents different starting point or trajectory

c. Nodes effectively explore different regions of solution space

d. Collision or convergence indicates potential solution

e. Exploration parallelism scales with network size

0024 Application: Distributed Random Selection:

0025 For selection problems, MAC-seeded hashes provide distributed randomness:

a. Payload encodes selection parameters (candidates, criteria)

b. Each node's hash provides independent selection

c. Aggregation determines consensus selection

d. No single node controls outcome

e. Selection is reproducible given all contributing MACs

0026 Security Properties:

0027 The MAC-seeded approach provides several security guarantees:

a. Non-Manipulation: No node can control another's output without knowing its MAC

b. Verification: Any output can be verified given claimed MAC and payload

c. Distribution: Influence is distributed across all participating nodes

d. Determinism: Same inputs always produce same outputs – auditable

e. Privacy Option: MACs can be hashed before use, preserving privacy while maintaining uniqueness

0028 Efficiency Properties:

0029 The approach achieves efficiency through:

a. Single Broadcast: One message produces N results from N nodes
b. No Coordination: Nodes compute independently, no consensus protocol required
c. Parallel Execution: All nodes compute simultaneously
d. Minimal Payload: Only application data transmitted, seeds are local
e. Existing Infrastructure: Leverages existing network broadcast mechanisms

ENABLEMENT

0030 The inventions described herein are enabled at the system and protocol level. A person of ordinary skill in distributed systems, cryptography, and network protocols can practice the claimed methods using the architectural descriptions in these specifications together with any suitable software/hardware stack.

0031 A working, non-limiting embodiment is publicly available at:

https://github.com/zeam-labs/zeam-testnet

0032 Reference Implementation Components:

a. `Dispatcher` structure managing broadcast-collect-assemble cycle
b. `Dispatch()` method implementing full cycle with timeout and callbacks

c. `collectHashes()` method accumulating unique hashes from network

d. `BroadcastZEAM()` method sending payload through transaction pool

e. `OnTxHashReceived` callback receiving hashes from network peers

f. `DispatchResult` structure containing collected hashes and metadata

CLAIMS

1. A method for producing diverse computational outputs from identical inputs across distributed network nodes, comprising:
   a. preparing a computation payload containing application-specific data;
   b. broadcasting the payload to multiple nodes in a distributed network through peer-to-peer message propagation;
   c. at each receiving node, computing a cryptographic hash of the concatenation of a node-specific hardware identifier and the received payload, producing a node-unique output from identical payload data;
   d. propagating computed hashes back through the network to a collecting node;
   e. accumulating unique hashes from multiple nodes, wherein each hash represents a different computational perspective derived from the same payload; and
   f. assembling accumulated hashes into a composite result incorporating multi-perspective contributions from distributed nodes.

2. A system for collecting diverse computational results from distributed network broadcasts, comprising:

a. a payload encoder that prepares application data for network broadcast in a format compatible with peer-to-peer propagation;

b. a broadcast module that transmits encoded payloads to distributed network nodes through gossip protocols or transaction pool mechanisms;

c. a differentiation mechanism at each network node that computes cryptographic hash of local hardware identifier concatenated with received payload, ensuring identical payloads produce unique outputs across nodes;

d. a hash collector that subscribes to network hash announcements and accumulates unique hashes with deduplication;

e. a timeout manager that enforces collection duration bounds and supports early termination conditions;

f. a result assembler that combines collected hashes into composite outputs using application-specified combination logic; and

g. a verification module that can reproduce any node's output given knowledge of that node's hardware identifier and the original payload.

3. An apparatus for coordinating broadcast-collect-assemble cycles in distributed hash computation, comprising:

a. a state controller that manages dispatch lifecycle including idle, broadcasting, collecting, and assembling phases, preventing concurrent dispatch operations;

b. a deduplication store that maintains set of received hashes to identify and filter duplicate submissions;

c. a hash accumulator that stores unique hashes in both set and ordered list representations for flexible access patterns;

d. a collection timer that enforces maximum collection duration with configurable timeout values;

e. a callback interface that invokes application-provided hash processors for real-time filtering or early termination decisions;

f. a result packager that assembles collection metadata including hash count, duration, and peer statistics alongside accumulated hashes; and

g. a reproducibility interface that enables verification of any collected hash given the contributing node's hardware seed.

4. The method of claim 1, wherein the node-specific hardware identifier comprises a MAC address of a network interface, providing 48-bit globally unique identification.

5. The method of claim 1, wherein the node-specific hardware identifier comprises a composite hash of multiple hardware identifiers including MAC address, CPU serial, and storage device serial for enhanced uniqueness.

6. The method of claim 1, wherein the cryptographic hash function comprises keccak256, providing 256-bit output with cryptographic security properties.

7. The method of claim 1, wherein broadcasting comprises formatting the payload as a non-executing transaction and submitting to network transaction pools for gossip propagation.

8. The method of claim 1, wherein accumulating unique hashes comprises maintaining both a set data structure for deduplication and an ordered list for sequential access to received hashes.

9. The method of claim 1, wherein assembling accumulated hashes comprises XORing all collected hashes to produce a composite hash incorporating all contributors' perspectives.

10. The method of claim 1, wherein assembling accumulated hashes comprises selecting a single hash based on application-specific criteria including leading zeros, proximity to target, or index position.

11. The system of claim 2, wherein the differentiation mechanism hashes the hardware identifier before concatenation, preserving uniqueness while preventing hardware identifier disclosure.

12. The system of claim 2, wherein the broadcast module utilizes blockchain peer-to-peer protocols including devp2p transaction gossip for payload propagation.

13. The system of claim 2, wherein the hash collector implements early termination upon receiving a hash meeting quality criteria specified by the application.

14. The system of claim 2, wherein the timeout manager implements settle time logic that waits for additional hashes for a configured duration after the first hash arrives.

15. The system of claim 2, wherein the result assembler supports multiple combination strategies selectable per dispatch including selection, XOR, concatenation, and voting.

16. The system of claim 2, further comprising a statistics module tracking dispatch duration, hash count, peer count, and network latency metrics.

17. The apparatus of claim 3, wherein the state controller rejects new dispatch requests while a collection is in progress, returning an error indication to the caller.

18. The apparatus of claim 3, wherein the collection timer supports both absolute timeout and relative settle time measured from first hash receipt.

19. The apparatus of claim 3, wherein the callback interface supports a hash processor function that returns a boolean indicating whether to continue collection or terminate early.

20. The apparatus of claim 3, wherein the reproducibility interface enables audit verification that any claimed hash was correctly computed from the stated hardware seed and payload.

ABSTRACT

0033 A framework for distributed computation producing diverse outputs from identical inputs through MAC-seeded cryptographic hash differentiation. Each network node computes a cryptographic hash of its unique hardware identifier (such as MAC address) concatenated with a broadcast payload, ensuring identical payloads produce unique outputs across nodes. A

dispatcher coordinates the broadcast-collect-assemble cycle: payloads are broadcast through peer-to-peer networks, receiving nodes compute and propagate their unique hashes, and a collector accumulates unique results with deduplication. The system achieves cryptographic diversity without randomness, enabling parallel solution space exploration, multi-perspective semantic computation, and distributed selection mechanisms. All outputs are deterministic and reproducible given knowledge of the contributing node's hardware seed.