Radium - The Universe-Scale Ledger Without Limits

Introduction

Radium is a high performance Layer One blockchain that focuses on speed and security, tightly implemented in approximately 121,941 lines of C and C++ code. It features on-chain governance through staked shareholders, native tokens, atomic swaps and automated market makers all with sub-second block finality.

Investment

Building a high performance, secure and scalable Layer One blockchain takes many years of research and development from a team of expert engineers. With this comes costs from tools, software, hardware, real-world testing expenses, infrastructure, etc. This project is completely self-funded with no Grants, Community Donations or Venture Capital involvement. Upon production launch 80% of the initial supply will be distributed via ICO and the remaining 20% will be allocated to the founders.

Economics

Issuance

The genesis block mints 46,621,289.408 credits for network and ecosystem bootstrap and 80 credits for the solo epoch leader's stake account. There is no hidden supply, unlocks or vesting schedules. The founders are allocated 20% of the initial supply, approximately 9,324,257.8816 credits and ICO participants are allocated 80% approximately 37,297,031.5264 credits. Block rewards start at 1 credit. Every 4,536,000 blocks the reward reduces by 1%. 10% of each block reward is deposited into the treasury and the remaining 90% is deposited into the block creators stake account. Credit issuance is finite and the total issuance shall not exceed 500,000,000 credits.

Fees

Fees are split into three parts and deposited to the block creators stake account, void account and treasury account. When splitting fees if there is a remainder it is deposited into the void account. Credits that enter the void account become indefinitely locked.

- Native Transfer - Variable
- Token Register - 80
- Token Transfer - 0.000024

- Token Mint - 0.000036
- Token Update - Variable
- Atomic Swap - Variable
- AMM Create Pool - 0.000027
- AMM Add Liquidity - 0.000027
- AMM Remove Liquidity - 0.000026
- AMM Swap - 0.000029

Stake delegation add, remove and update have zero fees, however there is a minimum of 80 locked credits to add and maintain a delegation.

Staking

The network operates through a Proof-of-Stake system, it is what drives consensus. Deploying a single-node-per-delegation model promotes network decentralization by requiring stake holders to deploy their own physical hardware. Staked delegates earn a portion of the block reward as well as applicable fees for their participation.

Governance

The network functions as a DAO (Decentralized Autonomous Organization). The treasury is self-funded from 10% of the block reward as well as a portion of network fees. This funding model provides a reliable source of capital for development, marketing, and other project needs without relying on outside investors, sponsors or grants. Staked delegates have direct voting rights on network changes, proposals, and treasury disbursements.

Tokens

Without the use of smart contracts or system programs the network implements native token registrations, mints, transfers and updates.

DeFi and DEX

Native credits and token credits can be traded peer-to-peer with low fees and no KYC.

Swaps

Atomic The system implements a generic mechanism for swapping native credits and token credits. This system allows swaps to occur off-chain and settled on-chain with great flexibility for 3rd-party developers. Swaps can be fully settled at extremely high speeds, usually in under 700ms. Any staked delegate can order, pair, finalize and submit swaps to a leader for on-chain processing. When a swap is executed the delegate receives a portion of the fees.

Automated Market Maker The system also implements on-chain automated market maker (AMM) liquidity pools for continuous decentralized trading between native credits and token credits.

Each AMM pool maintains reserves of two assets and issues liquidity provider (LP) shares representing proportional ownership of the pool.

Swaps against AMM pools follow a constant-product pricing model, where the product of the two asset reserves remains approximately constant. Trading fees fixed at 0.3% are collected and distributed to liquidity providers.

Cryptography

Random Number Generator

ChaCha20 is used in deterministic epoch-related schedule generation mechanisms. Mersenne Twister 19937 64-bit is used in various non-deterministic, non-critical sorting mechanisms.

Hashing

Blake3 is the primary hashing algorithm used throughout the system. SHA-256 is used once as part of the Schnorr Tagged Hash domain separation mechanism. SHA-512 is used once as part of the BIP-39 Mnemonic generation algorithm. Lattice16 is used to establish the ledger state hashes as well as the sole system-wide state hash.

Signatures

Schnorr is the primary and only signing and signature verification mechanism.

Encryption

The system does not use encryption of any kind.

Network and Consensus

An IPv6-only network backbone powers the consensus algorithms, combining a high-throughput UDP transport with a multicast-based dissemination model to efficiently propagate messages across the network. This foundation supports a simple, fast, and secure Byzantine fault tolerant single-phase, two-chain design, with consensus rounds paced by verifiable slot timing that collapses traditional multi-step coordination into a unified voting cycle and enables deterministic sub-second finality.

IPv6

By operating exclusively over IPv6, the network benefits from simplified packet processing and source-controlled fragmentation via Path MTU Discovery, elim-

inating in-network fragmentation and middle-box interference. The absence of Network Address Translation restores true end-to-end connectivity between nodes, reducing congestion and latency while enabling efficient multicast-based data distribution. IPv6's vast address space allows globally unique node identities, and its native support for IPsec provides a foundation for secure, authenticated communication at the network layer.

Key benefits:

- No more NAT (Network Address Translation)
- Auto-configuration
- No more private address collisions
- Better multicast routing
- Simpler header format
- Simplified, more efficient routing
- True quality of service (QoS)
- Built-in authentication and privacy support
- Flexible options and extensions
- Easier administration

Byzantine Fault Tolerance

At the consensus layer a single-phase, vote-to-all Byzantine Fault Tolerant design built around a two-chain pipeline. Each round has a designated leader to propose a block, but voting is never centralized: every delegate broadcasts its vote to the entire committee, allowing all nodes to independently observe quorum formation and aggregate proofs locally. This removes the need for multi-step leader coordination and eliminates entire classes of attacks, including silent leader and tail forking, since no single node controls progress or proof assembly. Proposals can flow continuously while commitment trails safely behind in a bounded pipeline, enabling fast, deterministic finality even under the presence of adversarial network conditions.

Slots

Governed by a difficulty-based verifiable delay that requires the leader to compute a CPU bound work before assembling a block, with the difficulty serving as proof of CPU throughput. Each proposal therefore carries a slot, a proof showing that the leader yielded to the transaction queue and operated within required minimum performance limits. In this way, the delay functions as a proof of performance rather than proof of work.

Transport and Routing

The transport layer is built directly on UDP, giving the protocol explicit control over packetization, timing, and delivery behavior without the constraints of connection-oriented transports. Messages are propagated using a deterministic

multicast-based routing model, where proposals are distributed through scheduled layers, reducing redundant traffic, and ensuring rapid, predictable dissemination. By decoupling transport mechanics from consensus logic, the system sustains high throughput and low latency while preserving clear, protocol-level guarantees on message handling and progression.

Together, these elements create a robust foundation for high-throughput applications, where low latency and irrevocable transactions are critical for real-world adoption.

Transaction Processing

The system deploys a parallel transaction processing unit where as transactions arrive from the network their signatures are verified in parallel across a pool of CPU cores. Each transaction is validated and checked against system state on parallel pipelines. The system, running on AMD EPYC™ 4565P CPU's proved on the Staging network to able to ingest up to 64,000 transactions per second before a block is to be assembled.

Data Storage

The system stores data as key value pairs or using key value separation to store binary large objects (BLOBS) directly to disk with pointers stored in LSM trees. By storing blocks and other large objects directly to disk we bypass database synchronization and achieve high IO/s. The system uses many independent databases allowing block commits to be parallelized to a high degree. The system organizes historical blocks and transactions into rolling epoch databases to mimimize data storage overhead.

Snapshots

Due to the systems fast block interval it is not possible for new incoming nodes to synchronize the blockchain from the beginning in a timely manner. Because of this nodes create full state snapshots every 72'000 blocks and retain two additional snapshots. Nodes wishing to join the network synchronize their state from one of these snapshots and then by syncing up to the chain tip by requesting blocks from a randomized set of nodes. Snapshots are downloaded manually over HTTP from a nodes snapshot service port.

Public API

Nodes optionally allow clients to connect to their API services in order to obtain information about and interact with the blockchain. The system deploys a JSON-RPC interface for queries and for pushing transactional data to the leader nodes. A WebSocket interface is available in which clients can subcribe to events using key/value filters.

Acknowledgements

We thank the anonymous reviewers and testers for their participation and input. Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta and Ittai Abraham for their work on HotStuff BFT Consensus. Kevin Lewi, Wonho Kim, Ilya Maykov, and Stephen Weis for their ideas on homomorphic hashing and rolling state. HandSolo for their ideas on Proof-of-Performance and verifiable slots timing. The late Claus P. Schnorr for his work on creating Schnorr groups. Cobie for the article entitled "New launches - private capture, phantom pricing" and "On the meme of market caps & unlocks". Mohit Lad, Ricardo Oliveira and the UCLA Computer Science department for their insight on IPv6-only networks. Anyone else we may have forgot.