FERROS

WHITEPAPER V1.0 — MARCH 2026

Ferros

Global Value Exchange
at the Speed of Thought

Author Ferros Labs

Status Pre-release

<1ms	$0.002	∞	0%
EDGE FINALITY	PER TRANSFER	THROUGHPUT CEILING	MEV EXTRACTED

CONTENTS

Ferros is a new kind of financial infrastructure — not a blockchain, not a layer-2, but something fundamentally different. It is built for the world that is arriving: one where value moves as fast as information, where autonomous agents transact alongside humans, and where the system's success never makes it harder to use.

Two parties exchange value directly — sub-millisecond, no consensus required. They settle the net through a shared DAG consensus layer with sub-second finality. Netting compresses thousands of operations into single settlement transactions. Every new participant adds capacity. There is no throughput ceiling.

This architecture shares more DNA with CLS Bank — the interbank settlement system that clears $6.5 trillion per day in foreign exchange — than with any cryptocurrency network. But unlike CLS, Ferros is permissionless, programmable, and designed for an economy where autonomous agents are first-class citizens.

Gas is paid in stablecoins. Costs are fixed. Agents and humans operate on equal footing. Scale is infinite.

01 Introduction

1.1 The Problem with Blockchains

Every major blockchain was designed around a single idea: global consensus on a shared state. Every transaction, no matter how small, must be ordered, executed, and agreed upon by every validator. This architecture is elegant for trustless computation, but it creates three structural problems for financial infrastructure:

Consensus is the wrong bottleneck. When Alice sends Bob $5, there is no reason for 1,000 validators to process, order, and confirm the transfer. The only parties who need to agree are Alice and Bob. Global consensus is useful for dispute resolution and final settlement — not for every individual transfer. Forcing every operation through consensus is like routing every phone call through a central switchboard.

MEV is architectural, not incidental. Miner/validator extractable value exists because transactions must be ordered. Any system with a single transaction ordering authority creates an implicit auction for position. Front-running, sandwiching, and priority auctions are structural consequences of the single-sequencer model — not bugs that can be patched away. As long as ordering determines outcome, those who control ordering will extract value.

Token-gas is a friction brake. On every chain where gas is denominated in a native token, the chain's success makes it harder to use. When ETH rose from $200 to $4,000, a simple transfer went from $0.50 to $10. Token holders want the price up; users want costs down. These are often the same people. The contradiction is unsolvable within the token-gas model. For autonomous agents making thousands of transactions per day, volatile gas costs destroy budgetability and make operations unpredictable.

1.2 The Bilateral Insight

> *The insight behind Ferros is that bilateral operations do not need global consensus — they need bilateral agreement. Two signatures. If both parties agree on the state, it is the state.*

Most value movement is bilateral. Alice pays Bob. A buyer fills a seller's order. An agent pays for an API call. Two counterparties settle their net position.

Consensus is needed only for settlement of the net result, dispute resolution, and operations that genuinely require global ordering.

This is how the most efficient financial infrastructure in the world already works. CLS Bank settles $6.5 trillion per day in FX markets not by processing every trade individually, but by netting bilateral positions and settling the remainder. The gross-to-net compression ratio in FX markets typically exceeds 95% — $6.5 trillion in gross trades nets down to a few hundred billion in actual settlement.

Ferros applies this model to decentralized infrastructure: instant bilateral coordination at the edge, shared settlement at the root.

1.3 Design Constraints

Six constraints govern every design decision. When constraints conflict, earlier constraints take precedence.

1. **Sub-millisecond bilateral finality.** Edge operations must complete in under 1 millisecond. Root operations must finalize in under 1 second.
2. **Horizontal scaling.** Throughput must scale linearly with the number of active edges. No global bottleneck.

3. **AI agents as first-class participants.** Deterministic costs, scoped delegation, persistent connections, autonomous execution — not afterthoughts bolted onto a human-centric design.
4. **Purpose-built for value movement.** Every primitive is optimized for moving value. This is not a general-purpose computer.
5. **Stablecoin-first.** Users pay fees in stablecoins at fixed dollar costs. No user is ever required to hold the native token.
6. **Decentralized.** No single point of failure, no permissioned access, censorship-resistant.

02

Architecture

Ferros consists of two layers with a clean separation of concerns.

```
+--------------------------------------------------------------+
|                        EDGE LAYER                             |
|                                                               |
|  Alice ⟵edge⟶ Bob        Carol ⟵edge⟶ Dave                    |
|  Alice ⟵edge⟶ Carol      Agent1 ⟵edge⟶ Agent2                 |
|  Agent3 ⟵edge⟶ API       Maker ⟵edge⟶ Market                  |
|                                                               |
|  Bilateral. Instant (<1ms). No consensus.                     |
|  Both parties sign. Netting compresses N ops → 1.             |
+--------------------------------------------------------------+
|                        ROOT LAYER                             |
|                                                               |
|  DAG consensus (50ms optimistic, 900ms BFT final)             |
|  Settlement batches from edges                                |
|  Batch auction clearing (MEV-free)                            |
|  Protocol-native order book (CLOB)                            |
|  Direct transactions (no-edge fallback)                       |
|  Dispute resolution & force-settlement                        |
|  Cudo program execution                                       |
|  Data availability                                            |
+--------------------------------------------------------------+
```

2.1 How This Differs from Blockchains

Ferros is not a faster blockchain. It is a different primitive.

DIMENSION	BLOCKCHAINS	FERROS
Basic operation	Transaction ordered by consensus	Bilateral agreement between two parties
Consensus scope	Every operation	Settlement only
Finality model	Global ordering → confirmation	Bilateral signing → instant, settlement → BFT
Scaling model	Make consensus faster (diminishing returns)	Add more edges (linear scaling)
MEV	Structural (ordering determines outcome)	Eliminated (batch clearing, no ordering advantage)
Privacy	All operations public	Edge operations private; root sees net change only
Throughput ceiling	Bounded by consensus	Unbounded (each edge adds capacity)

2.2 How This Differs from State Channels

Ferros edges share DNA with payment channels (Lightning, Raiden) but solve the problems that prevented state channel adoption.

DIMENSION	LIGHTNING/RAIDEN	FERROS EDGE
Capital model	Lock capital per channel	Auto-managed allocations from shared root balance
Routing	Multi-hop routing (NP-hard)	No routing — direct bilateral or root fallback
Opening/closing	Expensive on-chain transactions	Lightweight root registration
Netting	No netting — each payment discrete	Full bilateral netting — settle the net
Scope	Payment-only	General value coordination + programmable via Cudo
Failure mode	Route fails → payment fails	No edge → root handles it directly

The critical difference is capital management. Lightning requires users to manually lock capital into specific channels, creating a liquidity fragmentation problem. Ferros manages allocations automatically from a single root balance — the SDK sizes, tops up, and rebalances edge allocations invisibly.

2.3 The CLS Bank Analogy

CLS (Continuous Linked Settlement) Bank is the world's largest settlement system for foreign exchange. It settles $6.5 trillion per day using a bilateral netting model: member banks accumulate bilateral positions throughout the trading day, then settle the net at the end. Gross-to-net compression exceeds 95%.

CLS BANK	FERROS
77 member banks, permissioned	Permissionless participation
FX trades between banks	Any value movement between any two parties
End-of-day batch settlement	Continuous settlement (configurable frequency)
Centralized settlement engine	DAG consensus with BFT finality
Requires massive institutional capital	Auto-managed allocations from any balance
Not programmable	Programmable via Cudo

03

The Edge Protocol

3.1 What Is an Edge

An edge is a bilateral relationship between two parties, registered on the root layer. It enables instant value coordination without consensus. Both parties maintain a shared state that tracks their net position across all assets.

Every transfer on an edge is backed by locked funds at the moment it executes. There is no credit, no IOUs, no promises. A user's money is either available instantly or frozen safely — never taken without consent.

3.2 Edge State

```
EdgeState {
    party_a:        Address,
    party_b:        Address,
    seq:            u64,                        // monotonically increasing
    net_positions:  Map<AssetId, i128>,          // positive = A owes B
    allocations:    Map<(Address, AssetId), u64>, // locked on root per party
    sig_a:          Signature,
    sig_b:          Signature,
}
```

Every state transition increments the sequence number, adjusts net positions, and requires both signatures. No state transition is valid without bilateral agreement.

3.3 Edge Lifecycle

Registration. Either party submits a registration transaction to root; counterparty confirms. Root records the edge. No fee to register.

Allocation. The protocol auto-reserves funds from each party's root balance. The SDK manages allocation sizing based on usage patterns. Sophisticated users can configure allocations manually.

Active. Both parties exchange state updates via direct P2P connection. Every update increments the sequence number, adjusts net positions, and both sign. Transfers succeed only if the sender's allocation covers the amount. Both parties independently anchor `(edge_id, seq, state_hash)` to root approximately every second.

Settlement. Edges settle to root periodically — time-based, threshold-based, or on demand. Settlement moves the net position and reconciles allocations. The protocol charges `$0.002 × (new_seq - last_settled_seq)` — the accumulated per-transfer fees.

Closure. Both parties agree to close, or force-close after force-settlement resolution. Final positions settled, remaining allocations returned.

3.4 Netting

Netting is the core efficiency primitive. When two parties exchange value in both directions, the net settlement is far smaller than the gross volume.

NETTING EXAMPLE

Alice and Bob execute 10,000 operations over 5 minutes. Alice sends Bob 100 USDC, Bob sends Alice 80 USDC, Alice sends Bob 50 USDC... After 10,000 operations, the net position is +47 USDC. Settlement moves 47 USDC on root. One transaction. 10,000 operations compressed to 1 settlement.

The capital efficiency is remarkable. A small allocation (say 500 USDC per party) can support millions of dollars in gross bilateral volume, because netting keeps the net position small. This mirrors what CLS Bank achieves at institutional scale.

3.5 Periodic Anchoring

Both parties independently anchor their latest co-signed edge state to root every ~1 second. An anchor is a lightweight broadcast — proof of the latest state, not a settlement.

```
EdgeAnchor {
    edge_id:    B256,
    seq:        u64,
    state_hash: B256,
    sig_a:      Signature,    // both parties must have signed this state
    sig_b:      Signature,
}
```

Approximately 200 bytes per anchor. Batched into DAG vertices as metadata — no execution, no balance changes. Root verifies both signatures before accepting.

Anchoring prevents stale state fraud. Root always knows the highest valid sequence number for every edge, verified by both signatures. Any settlement attempt with `seq < highest_anchored_seq` or with a state hash that doesn't match the anchor at that sequence is rejected instantly.

3.6 Force-Settlement

Either party can initiate force-settlement at any time, for any reason — counterparty offline, non-cooperative, or simply wanting to exit.

1. Alice submits a ForceSettle request to root with her latest co-signed EdgeState.
2. Root verifies: both signatures valid, state hash matches anchored hash.
3. Root publishes the ForceSettle request on-chain.
4. Bob has 24 hours to respond:
    - If Bob responds with a higher-seq state: that state wins.
    - If Bob doesn't respond: Alice's state is accepted as final.
5. Settlement executes. Remaining allocations returned.
6. If Bob was non-responsive: a $1 force-settlement fee is deducted from Bob's allocation and paid to Alice.

SAFETY GUARANTEE

Funds are frozen, never stolen. The worst case is a 24-hour delay. The $1 non-responsive fee makes mass-ghosting attacks (opening thousands of tiny edges and disappearing) economically unviable.

3.7 Privacy

Root sees only the net change, not individual operations. If Alice and Bob exchanged $1 million gross but net to $50, root sees a $50 settlement. The intermediate operations are private between the two parties.

This is not a privacy feature bolted on after the fact — it is a structural consequence of bilateral coordination. The root layer never had the intermediate data in the first place.

04

Root Layer

4.1 DAG Consensus

The root layer uses DAG-based consensus derived from the Mysticeti protocol family. Each validator produces vertices containing transactions. Vertices reference 2f+1 parent vertices from the previous round. A vertex is committed when its causal history satisfies the 3-round commit rule.

Two finality levels:

LEVEL	MECHANISM	LATENCY	SAFETY
Optimistic confirmation	2f+1 validators acknowledge, backed by staked collateral	50–300ms	Validator slashed if reverted
BFT finality	3-round Mysticeti commit rule	150–900ms	Mathematically irreversible

Applications choose their finality level. CLOB fills, agent operations, and transfers use optimistic confirmation. Large settlements and high-value operations wait for BFT finality. Edge operations are unaffected — still sub-millisecond bilateral.

4.2 What the Root Layer Processes

The root layer handles six categories of operations:

1. **Settlement batches.** Co-signed edge states with net positions. Verify both signatures, check sequence against anchors, execute net transfer.
2. **Direct transactions.** Value transfers between parties without an edge. Same $0.002 fee. The root layer is always available as a fallback.
3. **Batch auction clearing.** SwapIntents collected per block per pool. Uniform-price clearing on the constant-product curve. All participants get the same price. Ordering-independent — proven by test. MEV is structurally impossible.
4. **Order book matching (CLOB).** Protocol-native matching engine with price-time priority. Market rules defined by Cudo programs. Per-market parallelism.
5. **Force-settlement adjudication.** 24-hour response windows, anchor history verification, slashing for stale state submission.
6. **Cudo program execution.** Deterministic bytecode with declared contention domains. Programs call protocol intrinsics for value movement.

4.3 Batch Auction Clearing

MEV elimination is not achieved through clever ordering or encrypted mempools. It is achieved by making ordering irrelevant.

All swap intents for a given pool within a block are collected and cleared at a single uniform price determined by the constant-product curve. Every participant receives the same price per unit. There is no advantage to being first, last, or anywhere in between.

> PROPERTIES
>
> **Ordering-independent:** same result regardless of intent ordering. **Uniform price:** all fills at the same price per unit. **MEV-free:** no advantage to ordering, front-running, or sandwiching. **Batch efficiency:** 100 swaps produce 1 reserve update.

4.4 Protocol-Native Order Book

For markets where continuous price-time priority matching is the right model (derivatives, perpetuals, options), Ferros provides a protocol-native CLOB. The protocol owns the matching engine; market-specific rules are defined in Cudo programs that the matching engine calls into.

Traders open edges with exchange markets for instant order management:

```
Trader with edge to perps market:
  Submit limit order    ⟶ edge state update, <1ms
  Cancel order          ⟶ edge state update, <1ms
  Fill notification     ⟶ edge state update, <1ms
  P&L settlement        ⟶ periodic, compressed via netting
```

A market maker doing 10,000 orders per day settles a handful of net P&L transfers. All matching and margin validation executes on root — edges are for order management only. No "outrunning the oracle" on an edge.

05

Cudo: A Language Built for Value

5.1 Why a New Language

Solidity was designed for general-purpose computation on a world computer. It has no concept of contention, no awareness of where state lives, and no way to express that two operations on different accounts can safely run in parallel. Every contract is a black box to the infrastructure.

Cudo is a programming language designed specifically for value movement. It makes three things explicit that Solidity leaves implicit:

1. **Contention** — what state does this program touch, and does it conflict with other operations?
2. **Value flow** — where does money move, and under what conditions?
3. **Execution context** — should this run bilaterally on an edge or through global consensus?

5.2 Contention Domains

Every Cudo program declares its contention domain — a compile-time statement that tells the infrastructure exactly how it interacts with shared state.

```
owned     -- only the owner interacts. No ordering needed.
per<K>    -- partitioned by key K. Parallel across partitions, sequential within.
global    -- all operations must be ordered. Sequential.
```

This is not an annotation. It is a language-level construct that the compiler verifies.

5.3 Automatic Layer Routing

> *The programmer does not choose a layer. The language tells the infrastructure where to run. This is the architectural moat.*

CONTENTION	LAYER	LATENCY	PARALLELISM
`owned`	Edge (bilateral)	<1ms	Fully independent per edge
`per<K>`	Root (partitioned)	~500ms	Parallel across K values
`global`	Root (sequential)	~500ms	Sequential (ordered)

No manual sharding. No choosing an L2. No bridging between execution environments.

5.4 Protocol Intrinsics

Programs call protocol intrinsics — `transfer()`, `escrow()`, `schedule()`, `standing_order()` — that execute at native speed. The program handles *logic*. The protocol handles *value movement*.

INTRINSIC	WHAT IT DOES
`transfer(from, to, amount)`	Move value. Atomic, native speed.
`escrow(from, to, amount, condition)`	Hold value until condition is met or timeout expires.
`schedule(block, callback)`	Execute a function at a future block.
`standing_order(to, amount, interval)`	Recurring automated transfer.

5.5 Example: Revenue Split

```
contract RevenueSplit {
    contention: owned

    state {
        recipients: Vec<(Address, u16)>,
        owner: Address,
    }

    op receive_and_split() {
        let total = msg.value;
        for (addr, bps) in recipients {
            let share = total * bps / 10000;
            transfer(self, addr, share);
        }
    }
}
```

This program has `owned` contention — it runs on the edge, sub-millisecond, with no root interaction. Incoming payments are split and distributed instantly using protocol intrinsics.

5.6 Example: Lending Pool

```
contract LendingPool {
    contention: per<address>    // each borrower is independent

    op deposit(amount: Reference6) {
        transfer(caller, self, amount);
        deposits[caller] += amount;
    }

    op borrow(amount: Reference6, collateral: Reference6) {
        require(collateral ≥ amount * 150 / 100);
        transfer(caller, self, collateral);
        transfer(self, caller, amount);
    }

    op liquidate(borrower: Address) {
        let b = borrows[borrower];
        require(b.collateral < total_owed * 120 / 100);
        transfer(self, caller, b.collateral);
    }
}
```

`per<address>` means each borrower's operations run in parallel on root. A lending pool with 10,000 active borrowers has 10,000-way parallelism.

06

Value Movement Primitives

All of the following are protocol-native operations — not smart contract implementations. They execute at native speed without VM overhead.

Token Transfer. First-class value movement with `TransferAsset::Auto` (protocol resolves which stablecoin) and `Reference6` amounts (currency-agnostic fixed-point). Running balance tracker prevents double-spending within a batch.

Standing Orders. Recurring automated transfers that execute without user interaction. Auto-cancel after 3 consecutive failures. Agents can set up standing orders and go offline.

Scheduled Execution. Future-block callbacks with optional recurrence. Enables vesting schedules, periodic rebalancing, automated liquidation checks.

Conditional Transfers (Escrow). Protocol-native escrow with sender-revocable, mutual release, and arbitrated conditions. Timeout triggers automatic refund.

Multi-Party Settlement. Atomic settlement across multiple edges or parties. All-or-nothing execution.

07

Agent-Native Design

7.1 Why Agents Need Different Infrastructure

Autonomous software agents are not slow humans with faster fingers. They are a structurally different kind of participant: they transact at machine speed, need deterministic costs, require scoped authority, need persistent connections, and must operate autonomously. No existing blockchain was designed for this workload.

7.2 Agent SDK

```rust
// Connect -- just need stablecoins, no native token
let ferros = Ferros::connect(rpc_url).await?;

// Open an edge -- automatic allocation, invisible to the developer
let edge = ferros.open_edge(counterparty).await?;

// Instant bilateral transfer (<1ms)
edge.transfer(USDC, 100_000_000).await?;  // $100

// Standing order -- runs automatically, agent can go offline
edge.standing_order(StandingOrder {
    to: savings, asset: Auto, amount: ref6(100_000_000),
    interval_blocks: 7200, max_occurrences: None,
}).await?;

// Delegate scoped authority to a sub-agent
edge.delegate(DelegationScope {
    delegate: sub_agent_key,
    max_per_tx: 100_000_000,       // $100 per transaction
    max_daily: 1_000_000_000,      // $1,000 per day
    allowed_ops: vec![Transfer, SwapIntent],
    expires_block: current + 100_000,
}).await?;
```

7.3 Agent Capabilities

CAPABILITY	MECHANISM	WHY IT MATTERS
Persistent connections	Live edges with counterparties	No polling — state changes pushed in real-time
Autonomous execution	Standing orders at edge level	Agent doesn't need 100% uptime
Scoped delegation	Protocol-enforced spending limits	AI agent can spend $100/day without keys to $1M
Intent-based interaction	Declare outcome, protocol handles execution	Agent doesn't need AMM math knowledge
Deterministic costs	Fixed fee schedule	Agent knows exact cost before executing
Bilateral speed	Edge transfers in <1ms	Fast enough for real-time multi-agent coordination

08 Economics

8.1 Core Principle: Users Never Need FRS

Stablecoins are first-class. Users pay fees in USDC, USDT, DAI, or any protocol-registered stablecoin at fixed dollar costs. No user is ever required to acquire, hold, or interact with the native token FRS.

> *Every chain that forces token-denominated gas will lose the agent economy to chains that don't.*

8.2 Fee Schedule

Three fees. All flat. All in stablecoins. All known before you press send.

OPERATION	FEE
Transfer (edge or root)	`$0.002`
Swap / CLOB intent	`$0.025`
Cudo execution	`$0.01` + compute surcharge

No registration fees. No percentage-based fees. No congestion pricing. No gas auctions.

One price for transfers regardless of layer. The user doesn't know or care whether the protocol routed their transfer through an edge or through root.

8.3 The FRS Token

FRS is a fixed-supply, revenue-bearing governance token. 50 million supply. No inflation. No mint function after genesis.

Staked FRS participates in 80% of all protocol fees, distributed in stablecoins — real protocol revenue, not inflation disguised as yield. FRS is required to run a validator and to vote on protocol parameters. It is never required to send transactions, deploy programs, or use the network.

THE FRICTIONLESS FLYWHEEL

More users → more edge operations → more settlements → more fee revenue → higher FRS staking yield → more FRS demand → higher FRS price → more validators → more security → more users. **User costs: unchanged. Always $0.002 per transfer.**

09

Throughput and Performance

9.1 Edge Throughput

Each edge is independent — no shared consensus, no contention between edges. Throughput scales linearly.

NETWORK STAGE	ACTIVE EDGES	AVG OPS/EDGE/SEC	TOTAL OPS/SEC
Early	100	1,000	100,000
Growing	10,000	5,000	50,000,000
Mature	100,000	5,000	500,000,000
Large scale	**1,000,000**	**1,000**	**1,000,000,000**

There is no theoretical ceiling. Every new edge adds capacity. The root layer is a settlement rail, not a throughput bottleneck.

9.2 Root Throughput

DAG consensus processes settlement batches, not individual edge operations. The effective throughput is root TPS multiplied by the netting compression ratio.

9.3 Latency

OPERATION	LATENCY
Edge transfer (allocated)	**<1ms**
Edge order submit/cancel	**<1ms**
Root transfer (optimistic)	50–300ms
Root transfer (BFT final)	150–900ms
Batch swap clearing	50–300ms
Auto-settlement to root	1–5 seconds

9.4 Comparison

SYSTEM	THROUGHPUT	FINALITY
Ethereum	~15–30 TPS	~12 min
Solana	~1,500 user TPS	~400ms
Sui	~10K–100K TPS	~500ms
Visa	~65K peak	instant (authorized)
Ferros root	**100K+ TPS**	**50ms–900ms**
Ferros with edges	**unlimited (horizontal)**	**<1ms bilateral**

10

Security

10.1 Edge Security

Edges are self-validating through four mechanisms:

Bilateral signing. Every state transition requires both signatures. No party can unilaterally alter the shared state.

Locked allocations. Every transfer is backed by funds locked on the root layer. No credit risk. No IOUs. The worst case for any party is a temporary freeze, never a loss.

Anchoring. Both parties independently anchor their latest co-signed state to root every ~1 second, with both signatures required. Root rejects any settlement attempt with a sequence number below the highest anchor or with a state hash that doesn't match.

Force-settlement. Either party can exit at any time with a 24-hour response window. Funds are frozen during the window, never at risk.

10.2 Root Security

The root layer is secured by BFT consensus: $n = 3f+1$ validators, tolerating up to f Byzantine faults. All validators independently verify every operation. Formal safety proofs have been completed for the consensus mechanism.

10.3 Algebraic Verifiability

Unlike general-purpose VM execution, most operations on Ferros are structured math: netting settlement checks `sum(credits) == sum(debits)`, batch clearing verifies the `xy=k` invariant, transfers check `sender_balance ≥ amount`. These are cheap to verify. The system does not require heavy re-execution for the common case.

10.4 Proofs and Attestations

A co-signed edge receipt is a stronger proof than a blockchain transaction hash — it is the signed agreement itself, not a pointer to a record. Both parties attested to it. Neither can deny it.

Balance attestations include both assets and liabilities, preventing solvency overstatement.

11

Data Availability

The root layer provides data availability through Reed-Solomon erasure coding. Block data is encoded into n shares (one per validator) such that any k shares are sufficient to reconstruct the full data. Default parameters: k=8, n=22 for a 22-validator set. The network can lose up to 14 validators and still reconstruct any block's data.

12

Adoption Strategy

Phase 1: AI Agents. The primary adoption driver. The Agent SDK makes it trivially easy for any AI developer to add payments to their agent. Stablecoin gas, fixed fees, sub-millisecond bilateral transfers, scoped delegation, and autonomous execution.

Phase 2: Stablecoin Transfers. "Send USDC anywhere, $0.002, instant. No native token to buy." This competes with Wise, Western Union, and Stripe — not with Ethereum.

Phase 3: Trading. Batch clearing DEX (MEV-free spot) and CLOB markets (perps, derivatives). Edge-based order management gives professional traders sub-millisecond interaction.

Phase 4: Financial Ecosystem. Lending, credit markets, prediction markets, yield, structured products — all built on Cudo with protocol intrinsics.

13

What Ferros Is Not

Not a general-purpose computer. Ferros is purpose-built for value movement. NFTs, on-chain gaming, social networks, and complex composability chains are deliberately excluded.

Not a blockchain. There is no single chain of blocks. The edge layer has no chain — just bilateral agreements. The root layer uses a DAG, not a linear chain.

Not a layer-2 system. Ferros does not inherit security from another chain. It is sovereign infrastructure with its own validator set and BFT consensus.

Not a speculation machine. Ferros has no memecoin launchpad, no NFT marketplace, no incentive to gamble. The protocol is designed to move value, not to create new forms of betting on value. The native token exists to secure the network and earn protocol revenue — not to be the subject of leveraged trading.

14

Conclusion

The infrastructure that moves value should be as simple, fast, and predictable as the infrastructure that moves information. Two people exchanging value should not require global consensus any more than two people exchanging messages should require a central switchboard.

Ferros makes bilateral coordination the default path. Consensus is reserved for what actually needs it: settlement, dispute resolution, and operations that require global ordering. The result is a system where every new participant adds capacity instead of consuming it, where costs are fixed and predictable regardless of network success, and where autonomous agents operate as first-class citizens with deterministic economics and protocol-enforced safety.

> *The bilateral coordination model is not new — it is how the most efficient financial infrastructure in the world already operates. What is new is making it permissionless, programmable, and accessible to anyone with stablecoins and an internet connection.*

References

1. CLS Group. *CLS Settlement.* cls-group.com. Accessed March 2026.
2. Babel, Mavridis, Feng, et al. *Mysticeti: Reaching the Limits of Latency with Uncertified DAGs.* 2024.
3. Buterin, V. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.* 2014.
4. Poon, J. and Dryja, T. *The Bitcoin Lightning Network.* 2016.
5. Anthropic. *x402: HTTP Payment Protocol Specification.* 2025.
6. Bank for International Settlements. *Statistics on Payment, Clearing, and Settlement Systems.* BIS, 2024.

Ferros is open-source software under active development. The protocol specification, implementation, and all supporting documentation are available at the project repository.