IMPACTMESH

ImpactMesh Whitepaper

Hybrid Layer 2 Execution and Settlement for Real-Time Games

ImpactMesh is for games that want innovative ways to introduce decentralized, **open participation, open world-building, and open economies** without pushing ordinary gameplay through blockchain friction.

It is aimed at **real-time and simulated worlds** that need transparent rules, verifiable competitive outcomes, open server participation, and a way to settle value or resolve disputes on Layer 1.

The model is particularly attractive for games that want **AI agents and bots to participate** directly in the economy without relying on opaque operator privileges. That is especially relevant in PvP systems between autonomous agents, where the only credible way to ensure they are following the rules is to make rule execution provable.

TABLE OF CONTENTS

Why Build Games on Blockchains 01

The Challenges 02

How ImpactMesh Addresses Them 03

How the System Works 04

How to Build a Game on ImpactMesh 05

01 Why Build Games on Blockchains

Games are starting to use blockchains not because every action belongs onchain, but because some parts of a game benefit from stronger external guarantees than a conventional backend can provide. Persistent economies, competitive outcomes, shared world rules, third-party participation, and durable ownership claims all become more credible when they can be audited, replayed, and, when necessary, settled outside the operator's private database.

The practical question is therefore not whether a real-time game should run fully on a base layer. It is whether a game can preserve normal server performance while selectively gaining the benefits of open verification, external settlement, and broader participation where those properties actually matter.

The practical benefits of the model are straightforward:

- **Open world-building:** a universe allows the creation of adjacent worlds with independent but compatible rules.
- **Open economy:** assets and game-relevant state can be attested and settled externally, making economies easier to inspect, govern, and interoperate with broader crypto systems.
- **Fairness:** games can expose verifiable randomness, deterministic replay, and transparent rules rather than asking users to trust opaque server logic.
- **Low-friction blockchain connection:** a game can connect to a blockchain without imposing constant user friction or hard architectural lock-in, because settlement happens only when demanded.
- **Verifiable PvP matches:** competitive outcomes can be replayed, audited, and proven against deterministic rules and recorded history.
- **Games with stakes:** the system makes it possible to build games where outcomes have economic weight, because disputes and settlement can be escalated to Layer 1 when necessary.
- **Decentralized infrastructure:** server participation can be opened to additional nodes rather than remaining permanently closed under one operator.
- **Community-driven development:** a world can incorporate DAO-style governance or other onchain coordination mechanisms without forcing the full game loop onto a base layer.

Those benefits are only useful if they can be introduced without degrading the responsiveness and operational simplicity that real-time games depend on. That is the constraint that shapes the rest of the design.

In practical terms, ImpactMesh is aimed at games and interactive systems that need:

- fast authoritative execution,
- open and bridgeable economies,
- verifiable history,
- and a controlled path toward decentralized infrastructure, open participation, and stronger external guarantees.

These same properties also make the system unusually well suited to AI agents and bots. Autonomous participants work best when the environment exposes machine-readable rules, deterministic state transitions, signed actions, and replayable public history. In that setting, an agent does not need privileged backend access or brittle UI automation to operate. It can reason over explicit game state, act under the same rules as everyone else, and participate in economies, world maintenance, or competitive play in a way that is auditable after the fact.

That matters for more than convenience. If AI-driven participants are going to hold assets, trade, build, fight, or govern inside a persistent world, then their behavior must be legible to other participants and enforceable under shared rules. ImpactMesh gives developers a way to support useful autonomous actors as first-class participants in the economy rather than as opaque operator exceptions.

> NOTE
>
> **AI agents and bots can participate as first-class economic actors:** ImpactMesh gives them deterministic rules, signed actions, replayable history, and on-demand settlement, making them easier to trust, easier to govern, and easier to integrate into real game economies. In PvP between autonomous agents, that same structure makes rule execution provable rather than merely claimed, which is the only credible way to enforce the rules.

02 The Challenges

Blockchains are naturally good at maintaining ledgers, settling asset ownership, and providing credible external verification. They are not naturally good at running fast, state-heavy, low-latency games.

That mismatch is visible in games with frequent authoritative state transitions. A real-time strategy game, or any simulation that must operate at normal server speeds, cannot route every gameplay action through a conventional blockchain pipeline without paying a substantial cost in latency, throughput, and user experience.

The constraints are structural:

- **latency is too high** for real-time simulation,
- **per-transaction fees add unacceptable friction** to normal play,
- **global verification of every transition is too expensive** for high-frequency state,
- **general-purpose chains do not match spatially local execution well**, where most dependencies are local rather than global,
- **wallet and transaction flows introduce user friction** even before cost is considered, especially in games that require fast repeated interaction.

Taken together, these are not minor implementation annoyances. They are reasons why a normal real-time game cannot simply route its authoritative loop through a conventional blockchain stack and expect to preserve the qualities that make the game playable.

A purely centralized game server solves the performance problem, but leaves important problems unresolved:

- participants cannot independently verify authoritative state evolution,
- game economies remain operationally opaque,
- opening shared worlds to third-party operators requires trust in private backend infrastructure,
- bridging assets or state into major ecosystems still depends on trusting a single game operator.

That is the real tension ImpactMesh is addressing. The goal is to keep server-grade performance where games need it, while introducing verifiability, external settlement, and broader participation where those properties are worth paying for.

The practical problem is therefore not "how to put a game fully onchain." The real problem is how to combine:

- **server-grade execution performance**,
- **deterministic, replayable, and verifiable authority**,
- **credible ownership and blockchain settlement when needed**,
- **without imposing blockchain costs on ordinary gameplay**.

ImpactMesh is built around a simple premise: **decentralization has a cost, and games should pay that cost only where it creates clear value**.

03 How ImpactMesh Addresses Them

ImpactMesh approaches this as a **hybrid Layer 2** for high-performance interactive systems.

The core idea is to separate continuous game execution from occasional blockchain attestation and settlement. A developer can therefore run authoritative simulation with the performance of normal server infrastructure while still preserving the properties required for later verification and settlement to a chosen external chain.

The model rests on four design rules.

Deterministic authority

ImpactMesh assumes that authoritative execution must be deterministic. The same inputs must produce the same outputs.

This is what allows execution history to be treated as a verifiable chain of transitions rather than opaque server behavior. In practice, that means:

- deterministic state transition functions,
- explicit and protocol-derived sources of randomness,
- avoidance of non-deterministic runtime behavior,
- preference for integer or fixed-point arithmetic over floating point wherever transitions may need to be replayed, disputed, or proven.

The fixed-point requirement matters most for competitive and verifiable logic. Purely server-controlled PvE systems may tolerate floating point in isolated cases, but any PvP path, fraud-proof path, or proof-generating path should use deterministic arithmetic that can be replayed exactly across machines.

This also aligns with zero-knowledge execution. If a transition must later be proven rather than merely re-run, deterministic integer or fixed-point state transitions are the reliable foundation.

Local execution with a shared coordination point

Most game state is not globally coupled. It is local to regions, sectors, worlds, or neighboring entities. ImpactMesh takes advantage of that by dividing a universe into **cells**.

Each cell owns a part of the world state and advances through its own transition function. Cells depend primarily on local neighboring state rather than on a globally serialized machine.

Alongside these spatial cells, the system maintains one or more **stem cells**. Every universe has at least one stem cell, and in most cases one is expected to be sufficient. Stem cells carry the information that must be shared across the universe, such as:

- time reference,
- protocol-derived randomness,
- global coordination data,
- shared economic or control state where required.

This creates a structure closer to spatial simulation and distributed systems than to a single monolithic blockchain VM.

Paid finality on demand

ImpactMesh does not assume that every state transition should immediately inherit base-layer security.

Instead, the system records deterministic execution continuously, while **stronger decentralized attestation and external settlement are requested on demand**. When a participant wants to bridge or settle some part of the state externally, the registered and staked nodes for that universe attest to the relevant transition history and post the resulting commitment to the universe's chosen Layer 1 bridge target.

That bridge path is security-critical. It relies on staked attestors, fraud proofs, and Layer 1 enforcement rather than on informal trust in the operator.

That means:

- normal gameplay remains fast and inexpensive,
- users do not need to pay gas for every in-game action,
- developers can decide where the cost of decentralization is justified,
- external settlement is attached to demand rather than imposed on the entire simulation.

Gradual opening of participation

A world can begin under a familiar operating model: the developer launches a single server node, runs the universe, and defines the rules.

Those rules should be published in a deterministic form, as a Rust program that any participant can download, execute, and replay against the recorded chain history. If the rules and history are public, compatible third parties can verify the universe independently rather than trusting a private backend.

The same world can later open participation to additional nodes that subscribe to the protocol and stake through the Layer 1 bridge contract. That bridge contract acts as the node registry for attestation. This matters for both economy and world expansion.

Data availability follows the same broad intuition as modular systems, but with a simpler operational rule: required data is considered available if a participant can actually download the full data needed to replay the relevant history. Each node determines availability independently by succeeding or failing to fetch what it needs. That availability judgment is itself part of the verification process and, ultimately, part of attestation.

Why this model makes sense

ImpactMesh should not be presented as a claim that all game logic can or should be fully onchain today.

Its claim is narrower: some game properties benefit from blockchain-backed verification and settlement, but the real-time execution loop should usually remain off the base layer.

- **Compared with a traditional game backend**, it provides deterministic replayability, verifiable execution, and optional external settlement.
- **Compared with a general-purpose blockchain or ordinary L2**, it preserves the execution model needed for spatially local, high-frequency simulation.
- **Compared with fully onchain game designs**, it avoids forcing base-layer constraints into the real-time gameplay loop.

04 How the System Works

Universes, spatial cells, stem cells, and nodes

An ImpactMesh world is modeled as a **universe**.

A universe is composed of:

- **spatial cells**, which carry most local simulation state,
- **one or more stem cells**, which carry shared coordination state,
- **nodes**, which execute and validate the relevant parts of the universe.

The protocol treats participating operators uniformly as **nodes**. Nodes execute and verify the cells they participate in. If a node is the only participant covering a given cell or cell set, it can execute it directly. If multiple nodes overlap on the same cell set, they coordinate execution and verification through consensus.

All nodes in a universe must follow the stem cell layer. Beyond that, nodes participate in the spatial cells they cover. Coverage does not need to be evenly partitioned. Different nodes may run different numbers of cells, and overlapping coverage is supported.

Deterministic execution environment

The authoritative program for a universe must be distributable, replayable, and deterministic.

The intended execution model is compatible with Rust-based deterministic programs and WebAssembly-style runtimes. Regardless of the final implementation details, the operational requirement is that any participant should be able to:

- obtain the rules,
- obtain the execution history,
- replay the state transitions,
- verify that the observed world state follows from those rules and inputs.

Randomness must also remain deterministic from the protocol's perspective. In the current design, randomness is derived from the hash of prior stem-cell blocks, so that all honest nodes derive the same value from the same history.

Spatial sharding

ImpactMesh is designed around **spatial sharding**.

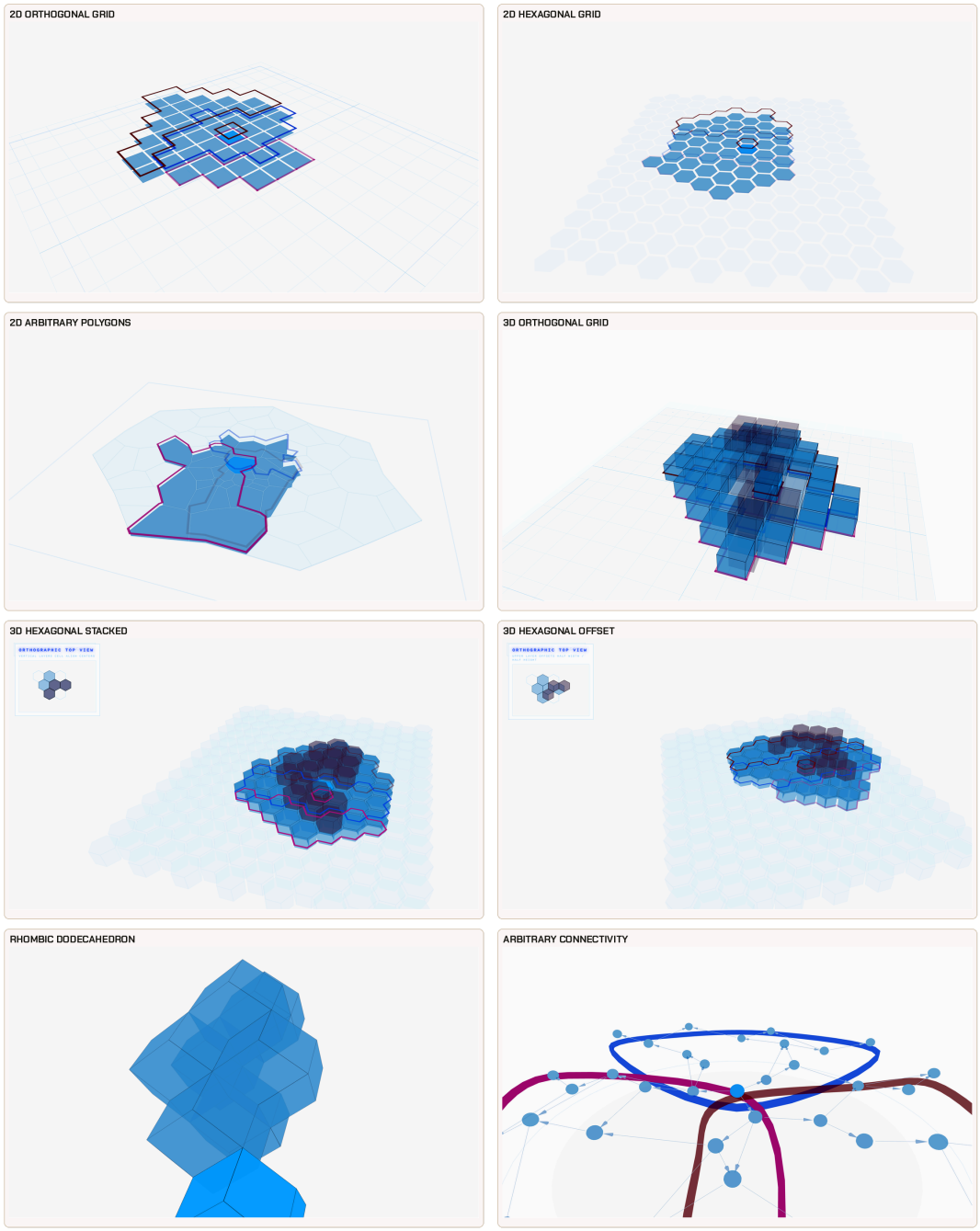
Cells may represent regions in 2D or 3D space, planets, sectors, application domains, or other spatially local partitions. The exact geometric packing is not the important part. The important part is the dependency model: a cell should compute most of its next state from:

- its own current state,
- explicit inputs,
- the state of neighboring cells,
- the shared information exposed by the stem cell layer.

This makes the execution graph predominantly local and allows throughput to scale with spatial decomposition rather than forcing all state into a single global pipeline.

Supported spatial configurations

ImpactMesh does not require a single canonical tessellation. The protocol is defined in terms of cells and explicit adjacency, which means different universes can choose the spatial organization that best matches their simulation, movement rules, and world structure.



The practical trade-offs differ by configuration:

2D orthogonal grid

A square grid is the simplest and most familiar option. It is well suited to simulation domains where movement, pathfinding, placement, and visibility naturally snap to axis-aligned cells. Its main weakness is that diagonal behavior requires explicit rules if distance and adjacency are meant to remain uniform.

2D hexagonal grid

A hex grid gives each cell six evenly spaced neighbors. That makes local adjacency more uniform than in a square grid and avoids privileged diagonals. It is a strong fit for tactical and territorial simulations where spacing and local maneuver should remain consistent in every direction.

2D arbitrary polygons

Some worlds are better represented as irregular regions than as uniform tiles. Arbitrary polygons are useful when geography itself carries meaning, for example in strategic maps, administrative regions, route networks, or terrain-driven partitions where cell size and shape should not be uniform.

3D orthogonal grid

A 3D orthogonal arrangement extends the square-grid model into volumetric space. It preserves the simplicity of aligned layers and direct vertical neighbors, which makes it appropriate for stacked simulation spaces, voxel-like worlds, and systems where vertical transitions should remain explicit and mechanically simple.

3D hexagonal stacked

A stacked 3D hex model keeps the in-layer advantages of hex adjacency while adding a simple layered vertical structure. This is useful when a world benefits from hex-like local movement in each plane but does not require dense cross-layer connectivity.

3D hexagonal offset

An offset 3D hex model shifts successive layers relative to one another. That produces richer cross-layer adjacency than a strictly stacked model and gives a more uniform treatment of vertical transitions and spatial diagonals. It is a better fit when the simulation should feel less like discrete floors and more like a connected volumetric field.

Rhombic dodecahedron

Rhombic dodecahedron packing is a more isotropic 3D partitioning model. Compared with layered 3D grids, it provides a more uniform neighborhood structure in volumetric space and is attractive for systems that want three-dimensional locality without privileging layer boundaries. However, it is also a more advanced and experimental choice because it is uncommon and less intuitive than familiar layered grids.

Arbitrary connectivity

Not every cell graph needs to be geometric in the conventional sense. ImpactMesh can also model arbitrary connectivity, where adjacency is defined by the application rather than by a regular tessellation. This is useful for systems organized around routes, relationships, influence graphs, portals, or other rule-defined topologies.

Overlap and boundaries

The protocol supports overlapping node coverage. Multiple nodes may execute and verify the same spatial cells.

Where overlap exists, the participating nodes coordinate those cells under Simplex. Each overlap set has a designated leader for the round according to the active leader-selection policy, and consensus determines the agreed transition history for that overlapped region. The specific policy is flexible; the important requirement is simply that each round has a designated leader.

Boundary communication is separate from overlap consensus. It is a network protocol between neighboring participants, and trust is determined pairwise through verifiable messages, replayable history, and the local rules each node applies when consuming boundary state.

Asynchronous local transitions and consume-only boundaries

Each cell defines its own state transition function. These transitions are **asynchronous by default**: a cell does not necessarily need a single global tick across the entire universe to make progress.

However, transitions are not unconstrained. Cells consume boundary state from neighboring cells, and the transition rules decide what to do when that boundary state is missing, delayed, or unresolved.

That means the decision to block is local to the cell logic itself. A given cell may choose to:

- block immediately until the required boundary state arrives,
- continue for some bounded period while using the last valid boundary state,
- switch behavior depending on the type of dependency involved.

Operationally, the model is:

- cells advance independently where possible,
- dependency edges are explicit,
- boundary consumption rules are part of the cell logic,
- stem cell progress remains the shared reference that all nodes must track.

This yields a system that is locally asynchronous, but still deterministic because dependencies, boundary-consumption rules, and ordering constraints are explicit.

The stem cell layer

The stem cell layer is the common reference that anchors a universe.

Every node must follow it. In the current design, stem cells are the natural location for:

- shared time reference,
- protocol-derived randomness,
- common global updates,
- shared economic control points,
- bridge-relevant state commitments.

In practice, this is the part of the system that behaves most like a conventional blockchain. It is the common coordination layer across a sharded execution mesh.

Continuous history and Layer 1 finality

Each node contributes to an ordered history of state transitions for the cells it runs and for the shared stem-cell progression it observes.

This creates a replayable chain of execution. The important distinction is that **replayability exists continuously, while finality is obtained only when the relevant bridge commitment is accepted on Layer 1**.

That distinction is central to ImpactMesh:

- the protocol continuously preserves the information needed for verification,
- but it only pays for wider attestation and external settlement when some participant needs it.

Consensus with Simplex

The coordination model is based on **Simplex**, a BFT protocol suited to partially synchronous networks.

At a high level, Simplex gives ImpactMesh a practical way to coordinate shared progression without relying on a separate complex view-change mechanism. Each view has a designated leader determined by the protocol's leader-selection policy, nodes vote on the proposed block, and when quorum is reached the block becomes notarized and can be finalized according to the protocol rules.

For ImpactMesh, this matters in three places:

- **stem cell progression**, where nodes need a shared order for the common coordination layer,
- **cell overlaps**, where multiple nodes coordinate execution over the same spatial region,
- **bridge attestations**, where the registered and staked nodes listed in the Layer 1 bridge contract must attest that a requested settlement is valid.

Under the usual 3f+1 Byzantine assumption, Simplex-style quorums rely on 2f+1 participation to make progress and finalize. That is the relevant quorum model for ImpactMesh as well.

Attestation, bridging, fraud proofs, and stake

ImpactMesh treats bridging as a request for stronger external guarantees.

When a participant (game developer, node operator, or player) wants to settle some part of the universe to its chosen Layer 1 bridge target, the network attests to the relevant history back to the most recent settled checkpoint. The result is then posted through the bridge contract on that Layer 1 as the basis for settlement. Once the attested commitment is accepted through the bridge on Layer 1, it is final.

The Layer 1 bridge contract is therefore the key external coordination point. It maintains the registry of participating attestor nodes, holds their stake, and verifies the quorum material required for settlement. In the first version, the contract's job is limited to node registration and quorum attestation handling rather than richer application execution.

Fraud proofs are punitive, not a separate rollback path for already accepted finality. Their role is to expose provable rule violations, such as signing or submitting an attestation for a transition history that does not follow from the published rules, committed inputs, and recorded chain history. If any participant proves such fraud on Layer 1, the offending node's stake is severely slashed and the prover is rewarded out of that slashed stake. In the current design, a successful fraud proof earns 1/2 of the slashed stake.

Beyond provable fraud, faulty, unavailable, or equivocating nodes can also be penalized operationally. A natural policy is temporary stake lock-up and exclusion from the next round or rounds of participation.

The bridged object is not limited to one asset class. The protocol is intended to support:

- **arbitrary state commitments**, and
- **token balances or token-related state**.

The broader architecture can support different settlement targets over time, but a given universe chooses one bridge target and is not itself multi-chain. Early deployments can target whichever external settlement chain best fits the product and integration requirements.

Zero-knowledge proofs and PvP games

Zero-knowledge proofs are an important part of the longer-term verification model, especially for competitive and adversarial game logic.

At a high level, the relevant statement is simple: given a deterministic rule program, a committed pre-state, explicit inputs, and shared randomness, a prover can demonstrate that the resulting post-state was computed correctly without forcing every verifier to replay the entire computation directly.

For ImpactMesh, this matters most in verifiable PvP interactions. A proof system can turn ”trust me, the server applied the rules correctly” into ”here is a compact proof that the deterministic transition or disputed outcome followed from the published rules and committed inputs.” That fits naturally with the system’s emphasis on deterministic arithmetic, replayable history, and economically meaningful disputes.

ImpactMesh can support direct PvP verification from the start by leveraging modern ZK systems. The main advantage, compared with proving directly on the settlement layer, is flexibility in proof-system choice and the ability to adopt newer proving systems without redesigning the full game around the constraints of a single onchain verifier. ImpactMesh is not constrained to succinct SNARKs alone. Without the same proof-size pressure that comes with storing and verifying the entire proof directly onchain, the design space is wider, which makes ZK considerably more practical for production games.

05

How to Build a Game on ImpactMesh

ImpactMesh is intended to support staged adoption rather than demanding full decentralization on day one.

1. Launch a universe

A developer launches a universe with its stem cell, chooses its single bridge target, and publishes the deterministic rule set.

At the beginning, this may be a single server node operated by the developer. That is acceptable. The point is not to force immediate decentralization. The point is to ensure that the world is built on rules and execution history that can later be replayed, verified, and settled externally.

2. Run a game without per-action user gas

Users interact through cryptographic identities and signatures, but normal gameplay does not require them to fund every action on a base layer.

This is a critical usability requirement. A lightweight wallet or equivalent identity layer is sufficient for:

- account control,
- signing actions automatically,
- holding internal assets,
- preserving ownership semantics when later bridging is requested.

This avoids the standard failure mode of blockchain games, where the mechanics of settlement degrade the game itself.

3. Open the world to additional nodes

Once a universe is operating, additional nodes can subscribe to the protocol, register and stake through the Layer 1 bridge contract, and participate in execution and attestation according to the universe design.

This allows a world to expand without requiring all world-building and operation to remain in the hands of the original developer. Depending on the application, cells may correspond to regions, planets, sectors, or application-specific domains. That makes it possible to open parts of a shared universe to external operators while retaining common rules and shared coordination through the stem cell layer.

4. Bridge or settle when needed and economically viable

When economic settlement, withdrawal, or external proof of ownership becomes necessary, the relevant state can be attested and bridged to the universe's chosen Layer 1 target.

This is the point where the system pays for decentralization and connection with the blockchain world. The cost is attached to the settlement event rather than to every gameplay action that preceded it, and finality is obtained only when the corresponding bridge commitment is accepted on Layer 1.

Typical settlement reasons include:

- withdrawing value to the chosen Layer 1 target,
- opening up an in-game token economy for trading and participation in the broader blockchain economy,
- settling shared state into a broader crypto ecosystem,
- proving ownership or commitments outside the original universe.