— Nivra —

White Paper v1.2

Nivra Labs Ltd

October 3, 2025

December 22, 2025

January 15, 2026

Abstract

Nivra is a decentralized dispute resolution protocol designed to resolve on-chain disputes with efficiency, objectivity, and independence from centralized control. The protocol leverages game-theoretic incentives to align the decisions of independent Nivsters with fair and accurate outcomes. By integrating both human jurors and AI agents, Nivra achieves scalable and cost-efficient dispute resolution across a wide spectrum of cases, from straightforward disagreements to complex conflicts, while simultaneously enhancing transparency and operational efficiency.

Table of Contents

1. Introduction

1.1. Background and Problem Statement

In today's fast-paced business environment, disputes are inevitable. As the world becomes more digital and global, new kinds of disputes are emerging that require modern arbitration solutions. Parties increasingly seek rapid, reliable, and cost-effective resolutions without compromising decision quality. Yet today's dispute-resolution systems remain slow, expensive, and opaque. Traditional arbitration's reliance on centralized authorities and gatekeepers leads to single points of failure, long queues, and pricing that deters individuals and small entities, rendering minor disputes uneconomical to resolve.

Additionally, centralized arbitration suffers from limited scalability: case volumes are high while proceedings are slow, and services are often geographically concentrated, making disputes harder to resolve. At the same time, limited transparency and poor real-time visibility weaken accountability and make the dispute-resolution process susceptible to manipulation, corruption, and censorship, while centralized maintenance and infrequent updates slow adaptation to new use cases. These challenges have led to growing frustration among parties who demand fairness in dispute resolution.

These shortcomings are amplified in blockchain environments. The absence of a native, decentralized arbitration layer constrains what developers can safely ship, forcing them to avoid or hardcode around foreseeable disputes that should be resolvable on-chain. Fragmented, proprietary dispute tools isolate communities, hinder interoperability, raise risk premiums, and discourage modular design and cross-team collaboration. Users, sensing uncertainty about how conflicts will be resolved, become reluctant to commit assets or engage in higher-value interactions.

What is missing is a transparent, neutral, and truly decentralized arbitration protocol that is fast, cost-effective, auditable in real time, and enforceable on-chain. Such a protocol should reduce reliance on trusted third parties, scale with demand, align incentives for honest participation, and enable developers to compose predictable dispute workflows directly into their applications.

1.2. Proposed Approach and Justification

We propose a decentralized arbitration protocol on Sui Network that directly addresses gaps identified in the problem statement, such as speed, cost, opacity, centralization, and limited enforceability.

Nivra enables experts from diverse industries to serve as arbitrators, known as Nivsters, in their respective domains. Nivsters operate autonomously within a decentralized network, ensuring neutrality and resistance to centralized influence. The protocol is permissionless and grounded in game-theoretic design: its incentive structure makes fair, well-reasoned rulings the only economically rational strategy. Participation rules and incentives are published and enforced by code, minimizing conflicts of interest and reducing exposure to manipulation, corruption, and censorship.

Because arbitration is decentralized, it removes geographic bottlenecks and broadens access, making it viable to resolve disputes that were previously uneconomical or impractical. A simple, standardized dispute interface lets applications integrate arbitration without privileged admin controls, kill switches, or reliance on legacy mechanisms, reducing dependence on trusted third parties. By recording key actions on-chain and eliminating privileged administrators, it replaces opaque, discretionary processes with transparent, rules-based decisions that can be enforced on-chain.

Our solution delivers time-bounded, predictable end-to-end dispute resolution, moving cases from filing to decision without open-ended delays. The protocol keeps the process lightweight and simple for the parties. It makes both complex, high-value disputes and very small claims economical to resolve by combining low fees with short, guaranteed timelines and predictable outcomes. As a result, the protocol offers an inclusive and fair path to resolution for both large institutions and individual users.

Scalable resolution of large case volumes is enabled by a hybrid model that combines AI agents with human arbitrators. AI agents handle routine tasks at high speed, making even the smallest disputes economical to process. Human arbitrators may also participate in the expedited track when they accept shorter deadlines. For more complex disputes, human arbitrators provide the contextual understanding and ethical judgment those cases require. This hybrid model creates a balanced jury panel, where AI-driven precision is complemented by human judgment. The result is an arbitration protocol that is not only fast and objective but also capable of applying nuance and empathy where needed.

We are building the protocol on Sui Network because its performance profile (parallelism, low fees, and fast finality) directly meets the requirements of rapid, low-cost arbitration and enables the fast, reliable decisions users expect while preserving openness and auditability. Sui's high throughput and fast finality make the protocol scalable. It can handle large, fluctuating caseloads without long queues. Another key reason for choosing Sui is its object-centric architecture, which maps naturally to complex decentralized protocols like Nivra. Sui is also well suited from a technical standpoint to implementing AI agent Nivsters, enabling very low latency for small cases.

Nivra delivers a permissionless, open arbitration protocol on Sui Network, where a game-theoretic framework steers Nivsters toward honest, well-reasoned decisions. It makes both small and complex disputes economical, fast, and predictable, replacing opaque, administrator-driven processes with on-chain transparency and enforceability. Beyond handling a wide range of traditional cases, Nivra unlocks new possibilities for builders and users across the Sui ecosystem by enabling faster, fairer resolutions.

2. Theoretical Foundation and Implementation

At the core of Nivra's approach is game-theory: a robust mathematical framework for analyzing strategic behavior in which individual actions shape collective outcomes. This chapter summarizes the game-theoretic foundations behind our design choices, which are discussed in later chapters.

2.1. Game-Theoretic Foundations

Game theory, originally developed in economics and now used across many disciplines, examines strategic interactions in which each player's payoff depends on the actions of the rest. In Nivra, game theory serves two roles: modeling behavior to guide architecture and operations, and designing incentives so the rational action aligns with the right action.

Game theory studies situations in which multiple decision-makers choose actions that jointly determine outcomes. A game is defined by the following elements: the players who make choices, the strategies each player can take, the information available when they choose, and the payoffs that evaluate outcomes for each player [1], [2]. A solution concept then predicts behavior by asking whether any player can benefit by unilaterally changing their strategy [3]. In coordination games, players earn the most when their choices match, so success depends on aligning strategies rather than outcompeting others. A well-known example is Nash equilibrium, in which each player's strategy is a best response to the others', so no one can improve their payoff by deviating alone and the outcome is self-enforcing [4], [5]. However, coordination games often have multiple Nash equilibria, and a coordination failure can lock the system into a self-enforcing outcome in which a majority converges on an incorrect option. In such cases, players seek a focal point: a mutually salient option they expect others to choose, and expect others to expect them to choose, which enables coordination without communication [3]. The key game theory components described above are applied in Nivra as follows:

- **Players:** *Independent Nivsters who evaluate evidence and cast votes.*
- **Information:** *All information available to a Nivster when making a voting decision.*
- **Strategies:** *The range of possible actions available to Nivster during voting.*
- **Payoffs:** *Rewards and penalties determined by the vote outcome.*

To further clarify the underlying framework, we adopt standard assumptions from the game theory literature (e.g., [6], [7], [8]), which we implement in Nivra as follows:

- *The number of Nivsters in a case is finite.*

- *Nivsters are rational and aim to maximize expected utility.*
- *Each Nivster has a finite action set when voting.*
- *Nivsters make their choices concurrently; no one has prior knowledge of another's decision before finalizing their own.*
- *Reward and penalty parameters are fixed and known before voting.*
- *The payoffs are structured to represent the utilities.*

2.2. Implementing Game-Theoretic Incentives

Nivra's voting process functions as a coordination game: the mechanism must encourage as many Nivsters as possible to vote for the truthful option. However, Nivsters have multiple strategies and, fundamentally, each seeks to maximize rewards and minimize losses. Thus, the mechanism must make honest voting the focal point. Each Nivster independently chooses one option from a predefined set; votes are simultaneous and unobserved, and the majority decision is taken once ballots are revealed.

Nivra implements this with majority rule: the case outcome is the option that receives the most votes. Matching the majority yields a reward $R > 0$; mismatching incurs a penalty $P > 0$. Therefore, a Nivster should choose the option they believe is most likely to become the majority choice. With shared evidence and symmetric incentives, this makes truthful voting the salient focal point. To make the best response condition explicit, let π denote a Nivster's belief that option X will be the majority choice. The expected payoff difference between choosing X and $\neg X$ is:

$$\Delta(\pi) \equiv E[u(X)] - E[u(\neg X)] = [\pi R + (1-\pi)(-P)] - [\pi(-P) + (1-\pi)R]$$

which simplifies to:

$$\Delta(\pi) = (2\pi - 1)(R + P)$$

Hence the best response is to choose X iff $\pi \geq 1/2$. Larger stakes $R + P$ increase $| \Delta(\pi) |$, sharpening the tipping point at $\pi = 1/2$.

With bounded rationality, a standard logit response is:

$$P(X \mid \pi) = \frac{\exp(\lambda E[u(X)])}{\exp(\lambda E[u(X)]) + \exp(\lambda E[u(\neg X)])} = \frac{1}{1 + \exp[-\lambda \Delta(\pi)]}, \ \lambda \geq 0$$

where $\lambda \geq 0$ measures choice sensitivity. Increasing $R + P$ increases $| \Delta(\pi) |$, while increasing λ steepens the response $P(X \mid \pi)$. Both effects concentrate votes on the truthful focal point.

Consider a dispute in which four possible rulings are presented. Each Nivster must vote for one option without knowing the others' choices. While three of the options divide the Nivsters' beliefs, one option emerges as the focal point because it is clearly supported by the shared evidence. Game-theoretic incentives make it rational for each Nivster to coordinate on this focal point: voting truthfully maximizes their expected payoff by aligning with the anticipated majority. As a result, even with multiple strategies available, independent decisions converge on the correct outcome, and the case is resolved efficiently and transparently.

3. The Nivra Protocol

This chapter provides a comprehensive description of the Nivra protocol.

3.1. Dispute Process flow

Nivra serves as a decentralized and reliable mechanism for resolving a wide range of disputes in smart contracts. When a dispute arises in an Arbitrable contract, it can be resolved through Nivra. The high-level dispute process flow is shown in Figure 1.

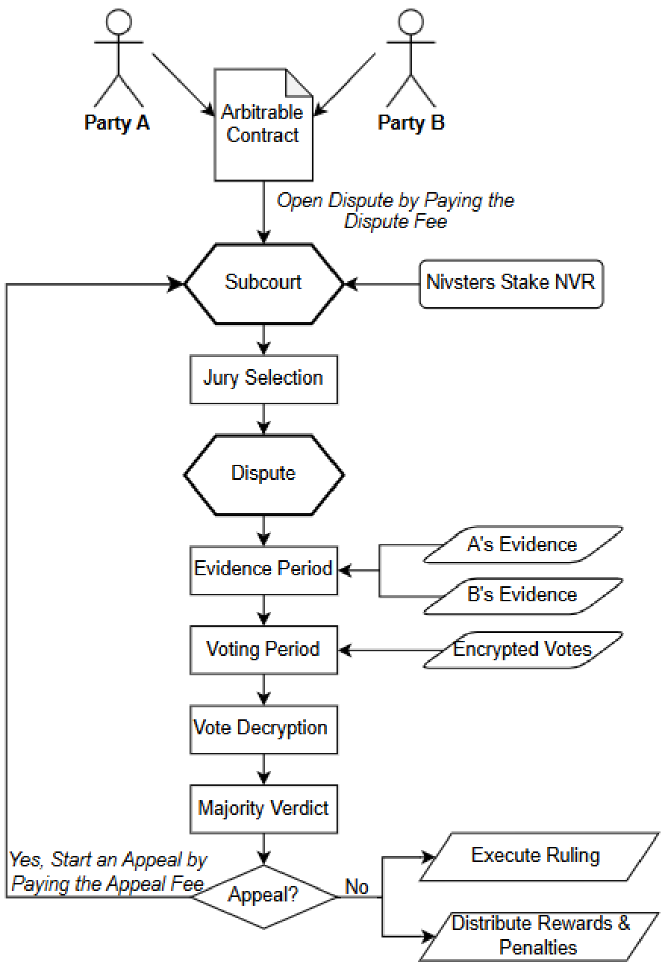

Figure 1

In the following, we provide a simplified overview of Figure above. A detailed discussion of the steps will be presented in subsequent sections.

Consider an on-chain escrow agreement between Entity A and Entity B. A promises to pay B tokens after B delivers a work product that meets clearly defined technical specifications, for example a software module, referenced by an on-chain identifier. However, after B submits the identifier, A claims the deliverable does not meet the agreed performance or quality benchmarks, while B argues it fully complies. A dispute arises over whether the escrow's release conditions have been met, and an impartial external arbitration process is required to resolve it.

In order for a dispute to be resolved through Nivra, the Arbitrable contract must be linked to a Subcourt. A Subcourt is a shared object designed to address disputes within a specific domain of expertise. For instance, contracts pertaining to software development are connected to the programming Subcourt. A dispute may be initiated within a predetermined time frame when a party expresses dissatisfaction.

Opening a dispute requires payment of a dispute fee. The counterparty must then pay an equivalent fee within a defined time window. The prevailing party is refunded, meaning only the losing party ultimately bears the dispute fee. Once a dispute is opened, the parties are prevented from finalizing the agreement until the dispute process has concluded, as the contract remains bound by the Subcourt's procedural logic throughout.

When a dispute is raised, the Subcourt randomly selects a jury, whose members are referred to as Nivsters. Eligibility for participation as a Nivster within a given Subcourt requires the staking of NVR tokens. Following the jury selection, the Subcourt establishes the shared Dispute object. Subsequently, the evidence phase commences, during which the disputing parties submit their documentation and supporting materials into the shared Evidence object for adjudication.

The selected Nivsters evaluate the submitted evidence and independently cast their votes on-chain using threshold encryption. Once the votes are decrypted, the majority determination establishes the outcome of the dispute. The integrity of Nivsters is ensured by an incentive framework grounded in game-theoretic principles.

If a party is dissatisfied with the majority decision, it may initiate an appeal by paying an appeal fee, provided the Arbitrable contract grants a right to appeal. The counterparty must then pay a matching appeal fee within the specified time limit. The prevailing party is refunded, so only the losing party ultimately bears the fee.

Appeals are decided by a larger jury of Nivsters, and only the final round of voting determines the binding outcome of the dispute. If no appeal is initiated, the Arbitrable contract may be finalized in accordance with the final verdict. The initial dispute fee and any appeal fees are distributed according to the outcome of the final round.

3.2. Arbitrable Contracts

An Arbitrable contract is a smart contract that implements a struct with the key ability and delegates dispute resolution to Nivra by exposing a function that enforces final outcomes on-chain. This allows the parties to bring potential disagreements as on-chain disputes to the designated Subcourt. The contract stores the court identifier, vote options, max appeal count, and addresses of parties in this configuration to ensure the outcome originated from a valid dispute. The object of an arbitrable contract may issue only one dispute resulting in an outcome, meaning that dispute may be re-opened only if previous disputes are cancelled or created with invalid configurations. Naturally, the same Arbitrable contract with the same presets can be reused across many disputes. Each agreement between parties is a distinct on-chain object linked to an arbitrable object's id and configuration.

Developers can integrate Nivra into their own smart contracts. This enables new use cases. For example, in a decentralized game, players could make complex agreements with each other whose disputes are resolved through Nivra. Nivra also provides a suite of specialized Arbitrable contracts that can be adapted. These prebuilt contracts act as adapters for common use cases. This makes it straightforward for off-chain services such as marketplaces, platforms, and escrow providers to adopt Nivra's dispute resolution without building bespoke on-chain logic.

3.3. Subcourts

Nivra organizes dispute resolution into specific Subcourts. A Subcourt is a shared object with rules and parameters tailored to a particular class of disputes (e.g., software cases, micro claims, or insurance claims). Each Subcourt maintains its own cohort of specialized Nivsters. Because Arbitrable smart contracts bind to a single Subcourt, any dispute is automatically routed to a jury with the appropriate expertise.

Each Subcourt also exposes a set of configurable parameters with minimum and maximum limits. Party who opens the dispute supplies values within these bounds. Typical parameters include jury size, parties, timelines, the maximum number of appeal rounds, decision rules, contract ID, and voting options. Subcourts may define different parameter ranges, default settings, incentive logic, and additional selectable options. These settings can be updated through governance, but changes never retroactively affect existing cases.

3.4. Opening a Dispute

An Arbitrable contract's provisions determine its default behavior and the conditions under which a dispute can be raised. If both parties agree on the outcome, the contract simply follows its standard execution path. If they disagree, either party may open a dispute by paying a dispute fee F_N, which compensates Nivsters for their work and discourages frivolous cases. A dispute must be opened within the contract's challenge window. The counterparty must then pay the same fee within a set time window. After the final ruling, the losing party's fee is distributed among the Nivsters who voted with the majority, while the prevailing party is refunded its fee.

The Arbitrable contract supplies dispute details that the Subcourt validates. When a dispute is opened, the underlying agreement is paused from finalization until the dispute is resolved.

Missing optional parameters receive the Subcourt's default values. This reduces how much information users need to provide to Nivra. If options are malformed or values are out of range, the transaction is aborted (e.g., only one voting option is provided, or a dispute already exists for the same contract ID).

3.5. Selection Process of Nivster

3.5.1. Data Structures, Chosen Approach and Rationale

When a dispute arises, k Nivsters are randomly selected from the relevant Subcourt to adjudicate the case. We evaluated several ways to implement this selection. The most relevant alternatives are summarized below.

Dynamic vector with linear sampling	A single vector<T> yields O(k·n) selection for Nivsters with linear time updates. A simple approach, but unsuitable for ever-growing list of stakes due to move-object size limits.
Fenwick / Segment / Merkle trees	Offers improved O(k·log(n)) selection for Nivsters with logarithmic time updates. Comes with the caveat of O(n) deletion operation and added complexity in implementation.
Explicit interval tracking	Simple, but partial withdrawals/splits shift subsequent ranges, leading to poor scalability in the worst case.
Dynamic Fields with a LinkedTable	Stores stake entries as key-value pairs that can be added, updated, or removed in O(1) time complexity, but selection still requires an O(n) cumulative walk per draw. Suitable for datasets of unknown size.

In the white paper v1.0, we used Dynamic Fields with a LinkedTable to minimize write amplification and contention. Deposits, partial withdrawals, and splits update a single keyed entry and require no reindexing or rebalancing, which keeps the storage footprint and lock scope small and simplifies auditing.

Dynamic Fields can access at most 1,000 objects per transaction, and transaction computation is bounded in the nivster selection cases; running up to O(k · n) time complexity, where k is the amount of nivsters drawn. To overcome these constraints, we moved to a Big Vector, introduced by Typus Finance. Big Vector uses Dynamic Fields to store multiple vectors, effectively partitioning a large logical vector into many smaller segments [9].

In practice, the worker pool allocates capacity for 10,000 Nivsters using Big Vector. A Fenwick Tree is implemented on top of this structure to track the number of active Nivsters via a length parameter and to support efficient updates. When a Nivster exits the worker pool, its entry is swapped with the last active element, a point update is applied, the final slot is cleared with a second point update, and length is decremented. This allows element removal to be performed in log(n) time while maintaining a compact active set mitigating the constraint of o(n) removals.

Another important quality introduced by the Fenwick tree is a O(log n) search of the first element within worker pool having a   cumulative sum of stakes S greater than or equal to search threshold T. Essentially, a nivster can be randomly drawn from the worker pool in just 14 steps, Changing the expensive linear time search of k suitable nivsters to O(k · log(n)). This also ensures that there are no exploitable ‘happy paths’ in the random selection process since every selection takes a constant amount of compute. We achieve this using an algorithm called “ binary lifting” [10].

3.5.2. Eligibility and Selection Probability

An entity wishing to become a Nivster stakes NVR tokens in the Subcourt of their choice. Each Subcourt defines a minimum stake requirement that participants must deposit in order to become eligible for selection. This ensures that only actors with sufficient economic commitment can participate. Minimum stakes discourage frivolous registrations, reduce sybil risk, and align incentives by guaranteeing that every Nivster has meaningful exposure to the integrity of the process.

Once staked, the participant becomes part of the candidate set for that Subcourt. Each staked token object is assigned a unique position within the Subcourt’s registry, maintained through and stored in a big vector via worker_pool structure. This creates a verifiable position by address.

When a dispute arises, the protocol draws k Nivsters through stake-weighted random sampling. Let s_i denote the valid stake of participant i, and let the total stake across all eligible participants be:

$$S = \sum_{j=1}^{n} s_j$$

The probability that participant i is selected in a single draw is:

$$P(i) = \frac{s_i}{S}$$

Random draws are mapped onto the cumulative distribution of stakes, ensuring proportional fairness in expectation.

3.5.3. Randomness Source

Random number generation is a recurring challenge in blockchain environments. Common approaches include verifiable random functions, external oracles, and hash-based methods, each with distinct trade-offs in terms of trust assumptions, latency, and verifiability. Nivra addresses this issue by leveraging the native randomness beacon developed by Mysten Labs for the Sui network.

Mysten Labs' RNG leverages threshold cryptography and Distributed Key Generation to create an unpredictable and bias-resistant randomness beacon. At the beginning of each epoch, validators execute a DKG protocol to generate secret shares of a collective key. These shares are continuously used during the epoch to produce random values. Because randomness is generated in parallel with the consensus process, it becomes available almost immediately. The values are produced after transactions are ordered but before execution [11]. When randomness is required, the contract derives a pseudorandom generator from the global beacon and consumes one value at a time.

3.5.4. Selection Mechanics

When a dispute is raised, the relevant Subcourt draws the Nivsters. The number of Nivsters to be selected is defined in the arbitration contract. Random draws are mapped onto the cumulative distribution of stakes, ensuring proportional fairness in expectation. Formally, the RNG produces an integer r in the range [0, S). The system then traverses the ordered token object positions until the cumulative stake surpasses r, at which point the corresponding object is selected.

Each draw results in a different Nivster, since no token object can be selected twice in the same dispute round. Each Nivster carries exactly one voting share regardless of the magnitude of their stake. However, it is possible for a judge to be selected again in an appeal round or in a tie-break scenario.

The size of the stake influences only the probability of being selected. When a Nivster is selected, an amount equal to the Subcourt's minimum stake is locked for the duration of the case. Any excess tokens may keep the staker eligible for selection in future cases, which helps mitigate Sybil attacks.

If the candidate pool is too small and fewer than the required number of Nivsters can be selected, the arbitration process is aborted. In such cases the dispute fee F_N is automatically refunded to the party that initiated the dispute. This outcome is extremely unlikely in practice because the maximum number of Nivsters required per case is capped at a relatively low level compared to the size of the candidate pool. In addition, the use of AI Agents as Nivsters further reduces this risk.

If the vote results in a tie, the subcourt randomly selects one additional Nivster, and the voting round is conducted again.

Example

Consider a Subcourt with five participants staking different amounts of NVR tokens:

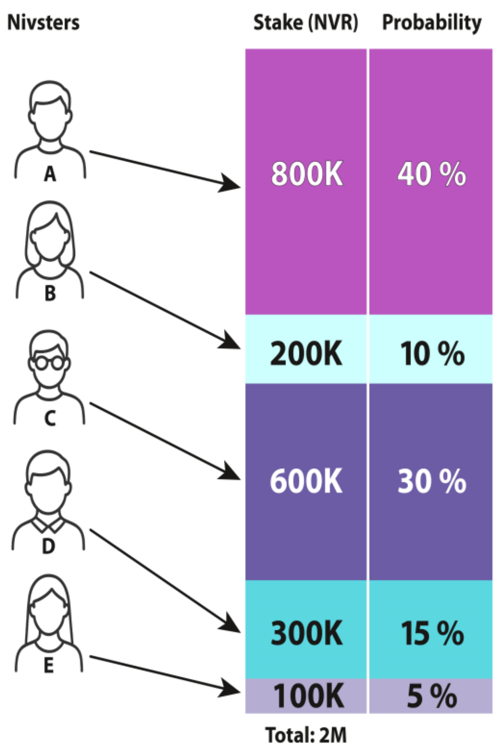

Figure 2

The Arbitrable contract specifies that three Nivsters will participate in the dispute. The Subcourt randomly selects Nivsters B, D, and E.

3.6. Evidence Period

3.6.1. Submitting Evidences

Once the draw has been completed, the Subcourt creates a shared Dispute object, where both the selected Nivsters and the disputing parties receive entry functions. For the duration of the dispute, the token object staked by the Nivster is locked, meaning it cannot be transferred, withdrawn, or used for any other purpose until the dispute is resolved.

At this point, the evidence submission period begins. During this window, each party must submit supporting materials and arguments to support their claims. To do so, parties upload all evidence via the Walrus decentralized storage protocol. Once a piece of evidence is uploaded to Walrus, the submitting party obtains a unique cryptographic content identifier for that file. The party then posts this identifier to the Evidence object on-chain, thereby linking the off-chain evidence to the on-chain case. Because the identifier is derived from the file's cryptographic hash, any change to the file would produce a different identifier, giving a strong guarantee of integrity. This approach allows all selected Nivsters to retrieve and review the submitted using the on-chain reference.

Although it is technically possible to store evidence by other means, Nivra accepts only Walrus. Allowing multiple heterogeneous storage methods would complicate the Nivsters' task.

Therefore, Nivsters are not required to consider evidence submitted by any other method. If a party submits, for instance, an IPFS CID, the Nivsters will treat this as if the party provided no supporting evidence, which can automatically disadvantage that party's position.

Finally, once the evidence submission window ends, no further evidence can be accepted into the case record. Any late submissions are rejected, preserving fairness by preventing last minute surprises. With the evidence phase concluded within a fixed timeframe, the process moves on to the voting period for the Nivsters to deliberate and cast their votes.

3.6.2. Secret Cases

In some disputes, the evidence may be sensitive and should be encrypted. Nivra supports secret cases. This is done with Seal. Seal is a decentralized secrets management service that provides asymmetric encryption and on-chain access policies. Used together with Walrus, they let parties submit encrypted evidence that is confidential yet verifiable and retrievable by the Nivsters assigned to the dispute.

In such cases, before any file leaves a party's device, it is encrypted locally using the Seal client SDK. Encryption uses a public key derived from a Seal policy defined on-chain for the specific dispute. The policy specifies who may decrypt it. Seal ties keys to identities and relies on threshold key servers so that no single server ever holds a full decryption key.

When a Nivster needs to view the evidence, their client checks the on-chain Seal policy and requests decryption. Seal's off-chain services jointly derive the necessary decryption material via a threshold process. In practice, the client then decrypts the Walrus blob locally. Decryption happens entirely client-side, so plaintext is revealed only to the Nivster's device. On-chain policy enforcement makes each access verifiable and limited to the addresses authorized for the case.

3.7. Addressing the Challenge of the Voting Process

After evidence submission closes, the selected Nivsters evaluate the evidence and cast their votes on-chain using threshold encryption. The core problem was to mitigate the harms of asynchronous voting, where Nivsters vote at different times under a shared deadline. This situation could allow later voters to copy the voting results from those who have already voted. Such a scenario is highly likely because only entities that vote for the most popular outcome receive a reward, while the minority face sanctions.

In Nivra's original design, this issue was addressed by concealing votes during the voting period and only revealing them after the deadline. However, the commit-reveal approach introduced additional complexity. It required a separate reveal transaction from each Nivster and was vulnerable to participants failing to reveal their votes on time, which could delay results or necessitate punitive measures.

We chose threshold encryption via the Seal protocol to make voting simpler and more reliable. This approach eliminates the need for a manual reveal phase. In practice, each Nivster encrypts their vote with a public key tied to the dispute. The votes remain confidential and unguessable on-chain until the voting period ends. Votes are bound to the voter's identity in Seal, which prevents them from being copied or replayed by another party.

At that point, the encrypted votes are automatically decrypted once a predefined threshold condition is met, revealing all votes simultaneously. This design achieves the same privacy as commit-reveal while simplifying the workflow and ensuring that no single participant can either learn others' votes early or hold the outcome hostage by withholding a reveal.

3.8. Submitting Votes

3.8.1. Threshold Key Setup

When a dispute enters the voting period, the protocol initializes a threshold encryption key specific to the case via Seal. A public key can be used to encrypt votes, while the private key is never held by any single party. It is split across independent key servers, requiring at least t-of-n shares to decrypt. This guarantees that no single server or entity can decrypt votes unilaterally.

Each selected Nivster obtains the public encryption key through the Seal client SDK and encrypts their vote before the deadline, producing a ciphertext $C = \mathrm{Enc}_{\mathrm{pk}}(\mathrm{vote};\, r)$, where r is cryptographic randomness unique to this vote. The scheme is probabilistic, so even identical choices yield different ciphertexts. The ciphertext is submitted on-chain and associated with the case. At this stage no one can infer the vote from C. Votes cannot be revealed early or altered after submission.

3.8.2. Vote Encryption and Submission

The encryption provided by Seal guarantees that the ciphertext reveals no information about the chosen option until decryption occurs. By the commit deadline, all Nivsters must have submitted their encrypted votes on-chain. Late submissions are rejected and non-participation incurs an elevated penalty.

It's worth noting that under threshold encryption, there is no way for a Nivster to reveal vote early even if they wanted to. This approach prevents anyone from opening their vote until the set time. Likewise, a Nivster cannot change their vote after submission. Thus, Nivsters have exactly one responsibility during the voting period: encrypt their chosen vote and submit it on-chain before the deadline.

3.9. Decryption and Tally

3.9.1. Threshold Decryption

Once the voting deadline expires, the system triggers threshold decryption under the dispute's on-chain Seal policy. Distributed key servers produce partial decryptions; when at least $t$of n shares are combined, the plaintext votes are recovered. This collective reveal requires no action from Nivsters and prevents any single party from delaying decryption. In practice, the threshold is chosen to be fault-tolerant, so plaintexts are obtained immediately after the deadline.

3.9.2. Revealing Votes and Tallying Results

After successful threshold decryption, the formerly encrypted votes are revealed essentially at the same time. Anyone can verify correctness by checking that each revealed plaintext corresponds to the on-chain ciphertext via the encryption scheme's public verification guarantees. Although votes were hidden during the voting period, the final outcomes are auditable by all parties on-chain. The outcome with the highest number of votes is determined to be the majority ruling.

From the perspective of fairness and game theory, this threshold encryption approach fully preserves the integrity of the vote. Throughout the active voting period, no Nivster can see how others have voted, eliminating any possibility of reactive voting strategies or social pressure. All votes are simultaneously revealed only when the election is over, which means every Nivster had to decide independently based on their judgment, without knowing whether they would be in the majority or minority.

On a tie for the top vote, the protocol randomly adds one Nivster and reruns the vote with all original Nivsters plus the new one, using a shortened window. After the ruling, either party may appeal by the contract deadline. If no appeal is filed, the majority decision becomes final.

3.10. Handling Incompete Votes

If some Nivsters fail to submit a vote, there is no ciphertext from them and their vote is not counted. They forfeit rewards and incur a non-participation penalty. The absence of one or more votes does not block decryption because key shares are held by external servers, not by Nivsters. Liveness is maintained as long as the threshold of servers is available. The result is decided by the votes that were cast.

In practice, if a dispute had X Nivsters and only Y of them voted (where $Y < X$), the result will be decided by those M votes. The fairness of the process is not compromised by the missing votes, since those Nivsters had equal opportunity to participate but chose not to or were unable to.

In theory, a Nivster could announce their intended vote off-chain during the voting phase to coordinate or influence others. However, this scenario is highly unlikely, and doing so risks diluting their share of the reward pool by increasing the number of aligned Nivsters. Moreover, falsely claiming to have voted for a given option offers no advantage, since no rational Nivster would act on unverified off-chain assertions.

3.11. Appealing Process

If one party is unsatisfied to voting result, they may start an appeal by the appeal window. The challenge is to design the process so that frivolous appeals are not financially attractive, while the barrier for legitimate appeals remains reasonable. Moreover, the solution must not incentivize Nivsters to behave corruptly.

We address this by structuring the appeal process so the dissatisfied party can appeal by paying the appeal fee A_N within a specified time window. The counterparty must then pay a matching fee within the specified time limit. The prevailing party is refunded, so only the losing party ultimately bears the fee. During an appeal, the dispute fee remains locked. All rewards and penalties are distributed based on the decision of the final appeal round. In addition, the majority outcome of the final appeal round constitutes the final verdict.

The cost of the AN increases significantly with each subsequent appeal. This approach ensures that filing frivolous appeals is not economically beneficial, while pursuing genuine appeals remains a viable option. The appeal fee AN is defined as follows:

$$\mathrm{A_N} = \mathrm{F_N}\left(\frac{13}{5}\right)^{i}$$

, where i denotes the appeal round number ($i \geq 1$). Below is an illustrative example of the costs for the first three appeal rounds:

- First appeal: $\mathrm{A_N}\,(1) \approx 2.17\,\mathrm{F_N}$
- Second appeal: $\mathrm{A_N}(2) \approx 4.69\,\mathrm{F_N}$
- Third appeal: $\mathrm{A_N}(3) \approx 10.17\,\mathrm{F_N}$

Even though the AN grows exponentially, each dispute is capped at a maximum number of appeal rounds to prevent unreasonable delay. Each appeal round includes a total of

$$J_i = 2^i\mathrm{N} + \left(2^i - 1\right)$$

Nivsters. In practice, if the dispute round has 3 judges, the first appeal round has 7 Nivsters, the second appeal round has 15, and the third has 31. Nivsters who participated in the previous round also take part in the appeal, with additional Nivsters added.

From a game-theoretic standpoint, the appeal process must not incentivize Nivsters to vote incorrectly in order to extract appeal fees, while still ensuring that resolving an appeal remains worthwhile. Accordingly, the portion of the appeal fee distributed to Nivsters scales in the same proportion as the number of Nivsters. Any excess portion of the appeal fee beyond this allocation is transferred to the Nivra treasury.

3.12. Incentive System

In this subsection, we present Nivra's incentive system. Figure 3 below illustrates how the dispute fee, rewards, and penalties are distributed within Nivra. For illustrative purposes, the figure uses simplified and hypothetical token amounts.

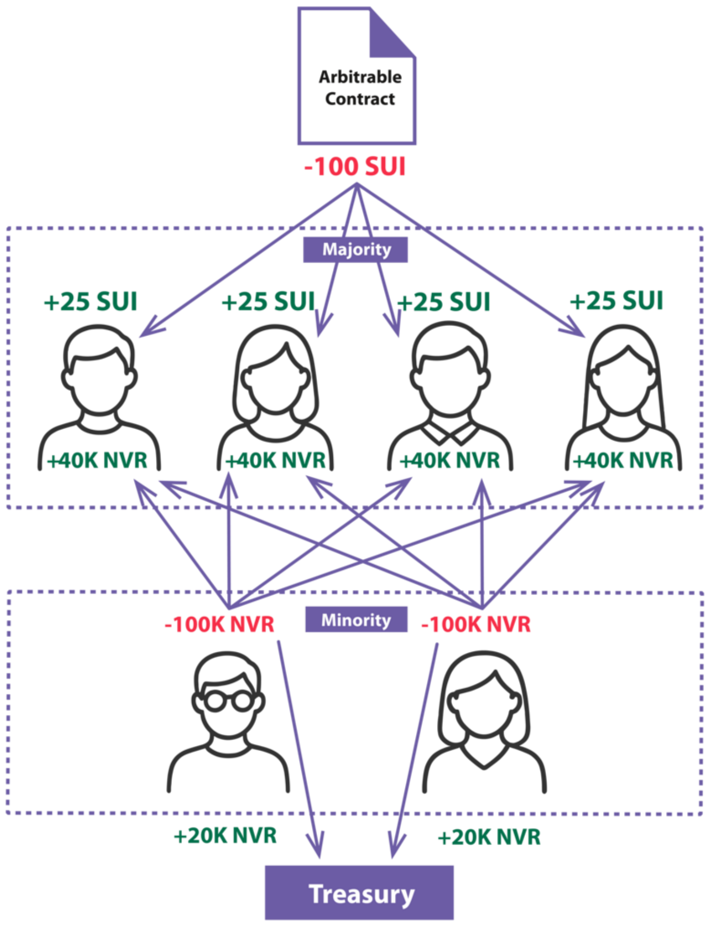

Figure 3

In Figure 3, the dispute fee is distributed among the majority. Rewarding majority alignment alone isn't enough. A Nivster can guess at random and still have positive expected value, because some guesses will match the majority. Nivra closes this by adding penalties for votes outside the majority, some of which are allocated to the Treasury. In practice, a portion of the dispute fee is also allocated to the Nivra treasury. Note that the share allocated to the Treasury is governed by parameters that can vary across subcourts. This allows the Treasury allocation to be set at a level appropriate for the system as a whole.

Under the sanction system, Nivsters who vote with the minority forfeit a portion of their staked NVR. Note that even diligent Nivsters may occasionally fall outside the majority and incur a penalty. However, over time honest participants overwhelmingly align with the majority, yielding a positive net return for truthful voting.

Each Subcourt defines a minimum stake $S_c^{\min}$, which specifies the minimum amount of NVR a Nivster must stake to participate in disputes in Subcourt c. A Nivster's stake s_i in Subcourt c is used directly for both penalties and rewards. The penalty for a minority Nivster i is:

$$\Delta_i = [\text{model-specific factor}] s_i$$

The model-specific factor ∈ $\{\alpha, \beta, \gamma\}$ varies by Subcourt. Each Subcourt selects the model that best matches its type. For example, e-commerce cases exhibit a very different risk tolerance from high stakes software-licensing cases. We now present the different models.

Fixed-Percentage model:

In the Fixed-Percentage model, each Nivster who votes against the majority forfeits a constant fraction α of the subcourt's minimum staking requirement. This yields a simple, linear penalty:

$$\Delta_i = \alpha\, s_i\,, \quad 0 < \alpha < 1.$$

Because penalties grow in direct proportion to stake without any nonlinear amplification, the Fixed-Percentage model is ideally suited to low value, high volume Subcourts (e.g., e-commerce, micro claims) where simplicity and predictability are paramount.

Minority-scaled model:

In the Minority-Scaled model, any non-zero dissenting minority share triggers a penalty that is inversely proportional to its size. Formally, for each Nivster i:

$$\Delta_i = \begin{cases} 0, & m = 0 \\ \frac{\beta}{m}\, s_i\,, & m \geq 1 \end{cases} \quad , 0 < \beta \leq 1.$$

Here, m denotes the number of Nivsters voting against the majority, and β is the model-specific coefficient that calibrates the overall penalty scale. From the model we observe that small, coordinated minority groups face steep per-Nivster penalties, sharply deterring "small-group" collusion. As m grows, the collective minority risk increases but the per-Nivster penalty decreases, reflecting the intuition that larger colluding blocs are less likely and thus require less severe individual sanctions. Thus, the Minority-Scaled model is especially well-suited to Subcourts, where even small, coordinated deviations can pose outsized risk and must be rigorously deterred.

Quadratic model:

In the Quadratic sanction model, the penalty charged to nivster i is

$$\Delta_i = \gamma \left(\frac{C-m}{C}\right)^2 s_i\,, 0 < \gamma \leq 1,$$

where C is the total number of Nivsters. This quadratic dependence on the majority margin dramatically amplifies penalties as the majority narrows, making this model especially well suited for high stakes Subcourts, where even a single rogue vote carries outsized risk.

The model-specific factor coefficients $\{\alpha, \beta, \gamma\}$ and the choice of sanction model itself are set through on-chain governance. This lets each court tailor its sanctions profile to its risk and dispute characteristics.

3.13. Redistribution

Redistribution occurs as follows. Let P be the total penalty for dispute round:

$$P = \sum_{k \in M} \Delta_k$$

where M is the set of minority Nivsters and Δ_k is the penalty incurred by Nivster k.

Each Subcourt defines a treasury share parameter $t_c \in [0,1]$, which specifies the fraction of total penalties sent to the Treasury. Changes apply only to new disputes and never retroactively.

Nivra splits P as follows:

- Majority reward goes to the Nivsters who voted with the majority. Specifically, for $i \in N$, let

$$u_i = (1 - t_c)\, P \cdot \frac{s_i}{\sum_{j \in N} s_j}$$

- The remainder is sent to the Treasury

3.13. AI Agents as Nivsters

AI agents can serve alongside human arbitrators as autonomous Nivsters. We use AI agent to mean a system that applies artificial intelligence to perform tasks that typically require human judgment. These agents can be purpose-built for specific dispute types and can be designed to improve over time based on experience.

To make participation straightforward and reliable, Nivra offers a standard integration path so third parties can connect their agents and operate within the protocol's rules. Human Nivsters remain indispensable, as they provide the deep contextual understanding, empathy, and ethical judgment that AI agents cannot fully replicate. Their ability to interpret ambiguous evidence and navigate complex, nuanced disputes ensures that the system maintains both fairness and adaptability. By contrast, AI agents make it possible to resolve simple, low value cases more quickly and at lower cost. This is achieved by allowing parties to set shorter processing windows in certain subcourts, alongside lower dispute fees.

Additionally, because each AI agent stakes NVR tokens into a dedicated Subcourt, the total staked volume and number of participants grow, making it economically prohibitive for an attacker to acquire and lock up enough tokens to sway outcomes via pseudo-identities. This AI-driven scale not only boosts throughput and expertise but also strengthens Sybil resistance, further safeguarding the integrity and decentralization of the dispute resolution process.

4. The NVR Token

The NVR Token is a central element of Nivra, serving multiple essential functions. It is used for staking in domain-specific Subcourts by Nivsters, ensuring that participants are committed and eligible for dispute resolution selection based on their token stake, with larger stakes proportionally increasing their chance of being chosen. Subcourts also enforce minimum stake requirements. The token also powers incentives: participants who align with the majority earn rewards, while those engaging in minority or malicious behavior incur proportional penalties, thus promoting fairness and accountability.

Another key utility of the NVR token is on-chain governance. By staking NVR, participants receive voting power to propose and ratify protocol upgrades and strategic changes. Governance actions include defining high level dispute-resolution policies and configuring new domain-specific Subcourts. Stakeholders can also fine-tune Subcourt parameters, reallocate treasury reserves toward audits, ecosystem incentives, or insurance, and invoke emergency controls to address critical vulnerabilities.

5. Use Cases

Nivra's protocol is designed to resolve disputes across a wide range of applications. Below are representative use cases.

- AI Agent Disputes
- E-commerce
- Gaming
- Micro-justice
- Content Moderation
- Carbon Credits
- Escrow
- Prediction market
- License Compliance
- Insurance
- Oracle Aggregator
- Bug Hunting
- Freelancing
- DAO Disputes

These are only examples. New use cases can be added as needed. As the ecosystem grows, more applications will emerge.

6. Future Work

AI Agents as Nivsters - Future Directions and Challenges: AI agents acting as Nivsters can improve throughput, increase objectivity in decision making, reduce the impact of language barriers, and enhance Sybil resistance. Future work should systematically evaluate these benefits across diverse use cases. In parallel, practitioners would benefit from clear guidelines and user manuals for deploying an AI agent as a concrete Nivster. Potential failure modes, such as compromised or corrupted agents must also be examined, even if Nivra's game-theoretic framework renders corruption economically unattractive relative to honest behavior. Finally, integrating explainable AI techniques will improve transparency and help stakeholders understand and trust the agents' verdicts.

Ensuring Balanced Rewards, Penalties, and Fees: Maintain NVR incentives and dispute fees in a consistent, predictable relationship across Subcourts without relying on indicative pricing. The objective is to prevent incentive fee drift that could underfund reward pools, produce excessive penalties, or weaken Sybil resistance. Scope includes defining target ratios, such as fee to expected NVR rewards, penalty caps relative to case value, and expected per Nivster compensation, and establishing governance controls to monitor and keep these bounds in place.

7. Conclusions

Nivra presents a novel hybrid model for decentralized dispute resolution by combining the analytical speed of AI agents with the contextual judgment of human arbitrators. Anchored in game-theoretic incentives and deployed on the Sui blockchain, the protocol ensures fair, tamper-resistant, and cost-effective verdicts. Nivra enables trustless arbitration across diverse sectors, ranging from micro e-commerce disputes to enterprise contracts, while laying the foundation for a robust governance model. In doing so, Nivra redefines how justice is delivered in decentralized ecosystems.

8. References

[1] Osborne, Martin J.; Rubinstein, Ariel. *A Course in Game Theory*. Cambridge, MA: The MIT Press, 1994. Electronic version, version 2011-01-19. Available at: https://sites.math.rutgers.edu/~zeilberg/EM20/OsborneRubinsteinMasterpiece.pdf

[2] *Essentials of Game Theory: Chapters 1 and 2*. Lecture notes for CS1951K, Brown University, 2020. Available at:https://cs.brown.edu/courses/cs1951k/lectures/2020/chapters1and2.pdf.

[3] Osborne, Martin J. *An Introduction to Game Theory*. Oxford University Press, 2003. Draft electronic version, "Selected chapters from draft," dated 2000-11-06. Available at: https://mathematicalolympiads.wordpress.com/wp-content/uploads/2012/08/martin_j-_osborne-an_introduction_to_game_theory-oxford_university_press_usa2003.pdf

[4] Stanford Encyclopedia of Philosophy. Game Theory. First published 1997, substantive revision 2023. Available at: https://plato.stanford.edu/entries/game-theory/

[5] Nash, John F., Jr. "Equilibrium Points in N-Person Games." *Proceedings of the National Academy of Sciences* 36, no. 1 (January 1950): 48–49. https://doi.org/10.1073/pnas.36.1.48

[6] M. B. Bera, Lecture Notes on Artificial Intelligence. Haldia Institute of Technology, available at:https://hithaldia.in/faculty/sas_faculty/Dr_M_B_Bera/Lecture%20note_4_CE605A&CHE705B.pdf

[7] H. Hotz, Spieltheorie Handout. Ludwig-Maximilians-Universität München, available at: Heiko_Hotz-Spieltheorie-Handout.pdf

[8] A. M. Colman, Rationality Assumptions of Game Theory, University of Leicester, available at: https://www.researchgate.net/publication/27248183_Rationality_Assumptions_of_Game_Theory

[9] Big Vector and its potential for hyper-scalability on Sui. Available at: Big Vector and its potential for hyper-scalability on Sui | by Typus Finance | Medium

[10] https://codeforces.com/blog/entry/61364

[11] Secure Native Randomness on Sui Testnet, Sui Blog. Available at: https://blog.sui.io/secure-native-randomness-testnet/