Tellor Layer

Draft Notice: *Tellor Layer is currently in production. This is an overview of the intended workings of the system. Feedback and contributions are very welcome and we hope to engage in a dialogue with the community throughout the design and build process.*

Last Edited: *14 Dec. 2023*

How the Chain Works

Tellor Layer is a stand alone L1 built using the cosmos sdk for the purpose of coming to consensus on any subjective data. It works by using tendermint to agree upon requested data and its values, and in cases where consensus is not reached, falls back to relying on an optimistic approach given reported values can be disputed.

Submitting Values

Any staked validator[1] can submit a value for a given query (i.e. data request, e.g. BTC/USD price). Once submitted for inclusion in a block, all other validators can add their value to the array of submissions.

All queryID's have an aggregation type associated with it (e.g. median, mode, average, etc.). When all reporters submit, the reports will be subject to weighted aggregation, the result of which will be the final value. For this reason, larger validators contribute more to the official value than smaller ones. Tips are distributed to all reporters for a given query using their weighted contribution.

Each official value takes three blocks (target time 1s per block). The first block is the tip (e.g. "I request BTC/USD"). The second block is the reporting of shielded values (just a hash of a signed value). This is to prevent mirroring (copying the first submission). In the third block, or commit phase, all values are revealed and must match the given shielded submission. They are also aggregated and an official value is determined for the queryID.

[1] Above a minimum stake amount and capped at a number X validators, where X is 100 to start with plans on growing

Validator Set Size
Chains using the tendermint consensus mechanism have a set number of validators. By setting the number of validators, this allows for efficient interconnectedness between chains as well as faster throughput. The tradeoff that any chain must consider is that more validators = slower blocktimes. Tellor will start with 100 validators (more than current reporting set), but will move to a larger set as technological advances or market conditions allow(e.g. no one needs sub-2 second blocks). The current target blocktime is 1 second.

Cycle List
In order to maintain freshness, governance will be allowed to vote on a list of enshrined queryIDs. If there is no tip in the previous block, validators will report for the next queryID in the list. This is to maintain freshness and ensure that tips do not need to be submitted in previous blocks to have a base level of reporting (thus avoiding wasting gas by validators who just want inflationary rewards). As long as time based / validator rewards are enough to cover the gas costs of reporting, the cycle list should be reported to consensus very often.

Disputing
Disputes work differently than with previous versions of tellor. Validator stakes are not tied to specific values, but rather just to a given reporter. For instance, a validator can submit ETH/USD once every second with their same stake. If they submitted a bad value 2 days ago, they can still be subject to a slashing event. Similar to the old tellor, however, is the idea that since you can censor a party by disputing, disputes are a bet or wager on who is correct (the slashed or the accuser).

Raising disputes is done by any party, but unlike the old system, stakers can use their stake (or part of it) to begin a dispute. Once initiated, the dispute fee and the potential slash amount (from the accused reporter) are put in escrow and removed from corresponding staking powers.

When initiating a dispute, the disputer will pick which category of dispute is in play. To do this, a party will submit a dispute against a given validator for one of three categories:

- Warning (1% of validator stake)- jail, but no lock, can call function to reenter. Note here that the minimum dispute is 1% of the minimum stake.
- Minor Infraction (5% of validator stake) - jailed for 10 minutes and out when release function called.
- Major Infraction (100% of validator stake) - jail until dispute over (obviously since 100% of stake)

A release function has to be called after a warning or minor infraction to ensure the validator looked at the dispute and implemented a fix as necessary. These infractions can generally be assumed to not be malicious.

After specifying the dispute category, the disputer will also submit an amount up to the minimum slashing amount before the dispute can initiate. If they don't have enough funds themselves,for up to one day, others can add to the slashing amount up to the max(1,5, or 100% depending on slashing category). Once the amount is hit (could be hit instantly upon proposing the dispute, or could take up to a day), the amount submitted for the dispute will be taken from the validator and placed into a locked stake.

For up to two days, stakeholders will now vote which side of the dispute they support. Layer will use the same voting distribution as traditional tellor governance:

25% users (tips)
25% validators
25% token holders
25% team

Once the two days are over there is a one day period where the dispute can be reopened and the same two day voting round is repeated. If at any point a quorum of >50% total voting power votes in favor of one side of the dispute, the dispute is considered finalized and no new rounds can be opened.

Once the dispute is resolved, the stake from the losing party is transferred into the stake of the winning party(ies). They are not released as free floating tokens so as disputes cannot be used as a way to exit staking faster, but rather transferred as undelegated, staked trb to the winning party(ies).

Note that each dispute round (even the initial one) takes a 5% dispute fee. Half of the dispute fee is burned and the other half is divided amongst voters in the system. The dispute fee then doubles each round up to the slash amount to further incentivize voting. Once quorum is reached on a dispute, further dispute rounds cannot be raised.

For usage purposes, values are not signed off on if the disputed reporter was the median contribution. Values that need to be flagged are also added to evidence when the dispute is called (an array of values). For this reason, users who rely on optimistic finality should be aware of dispute censoring attacks and the potential for values to take time to get through.

Tipping

All tips will be in TRB. Each queryID can be tipped directly and its tip will increase as more users tip it. Once a report is submitted and aggregated, 98% of the tip will be split amongst reporters for that queryID in that given block. There will be a 2% fee on the tips (to prevent vote farming/ spamming). This 2% fee will be burnt.

Note that tipping in Layer will consist of only one time tips. The idea of built in heart beats or price thresholds is nice, but the complexity added is unnecessary and better handled off-chain (e.g. tip bots like the autoTipper we currently have handles this functionality already).

Tokenomics

TRB will be the native currency of the chain and used for staking, tipping, and voting. All tokenomics will remain the same. 4k tokens will continue to be the dev share to the team each month and 4k tokens will be distributed to validators as inflationary rewards.

Gas fees will be given to the block proposing validator (selected randomly by validator weight), while tips and Inflationary rewards to validators are distributed proportionally to the oracle submissions based on weighted validator support. As an example, if a queryID is submitted for and gets 1% of the stake signing off on the value, this report will get some inflationary rewards, however if a more supported queryID gets submitted for with 50% of total stakes signing off on it, this will get 50x the rewards as the first value. There is no benefit being the only reporter for a query vs just signing off on one, unless of course there are tips that support one or the other.

The goal is to get broad support among as many queries as possible without incentivizing parties to report things that no one needs (e.g. report an obscure query that only I support) or disincentivize reports of new queries. Tips will still, and should, be used as the primary incentive mechanism used to garner support for a given query.

In order to be certain that block proposers include all reports for a given queryID, all parties that committed must reveal. If you commit and do not reveal, the block will finalize with ⅔ consensus. Inflationary rewards will be decreased by the same proportion as commits not revealed to disincentivize the proposer from excluding reveals. Additionally, proposers can be subjectively slashed for failing to include transactions in the commit phase should it begin to recur.

Usage

To use tellor layer data, a proof of a valid query is pushed to other chains. A light client bridge will be present on each chain and parties will relay the requested data and signatures as well as information related to validator set updates / chain upgrades. Relaying is a trustless role, however access controls can be put in place at the chain or user contract level.

Signing Prices

Once a value is aggregated on Layer, all validators sign the information. The resulting signatures will allow users to access the value, timestamp, and consensus threshold (amount of consensus signed off) of a given query.

Where: *data = value, timestamp, consensusThreshold*
Validators will:
sign(queryId, data, validatorNonce, powerThreshold, validatorSetHash, blockTimestamp)

Parties using layer can then grab the signatures and pull it over to their chain.

Users can also request signatures for previous blocks. E.g request(queryId, blockNo.) and it will return the same thing as before, but with the newer blockTimestamp. This is so parties can use optimistic values, by checking that the data has stayed on Layer for a certain amount of time without disputes, as data that has been disputed will not be signed.

Consensus usage
If ⅔ of tellor validators sign off on a value, it can be consumed instantly. A relayer will grab the desired data and signatures and push it to the consumer chain. Parties can then use the information in their protocol. There is no need to validate it any further, however waiting could be helpful, if only in extreme cases (e.g. large price moves), to check for forks, or widespread dispute conditions (e.g. a protocol or exchange failure/hack where api information may not properly reflect desired data specifications). In most cases however, this method will allow for updates as fast as the chain itself (e.g. 2 or 3 second data calculated from commit/reveal and block times).

Optimistic usage
Similar to consensus usage, parties will request signatures and grab data from the tellor chain, but should validate that the timestamp returned is a certain period older than the block timestamp signed. They can either use a very old value (similar to current tellor), and/or verify that X% of validators signed off on it.

Dispute Usage
Anytime parties are not using consensus values, they must be cognizant that the system is only as secure as the monitoring for disputes. If, for instance, a party requests an obscure piece of data that only one validator reports for, there will likely be no one checking this for disputes. For these reasons, if using an optimistic value, adding extra security can be beneficial. Running your own dispute monitor, educating more validators to support it, or even more secure measures such as using only if the median is within x% of a given (e.g. their own team's) reporter.

Best practices - consumer side dispute resolution / pauses
For parties using Layer, another option to increase security is consumer side disputes / pauses. For the pause, it could be the Maker design, where token holders of your own system can

freeze the system in the case of a bad value.[2] For disputes, parties could just design a system where it costs X dollars (a large amount) to delay the result. While delayed, they could initiate a dispute on the tellor system to remove a reporter or evaluate the situation. This would work well for systems that can handle delays (e.g. a prediction market delaying payouts). Just note that this would be on a per-user basis and custom as the cost to dispute/pause is also the cost to censor if it freezes the system for certain protocols.

Optimistic as consensus fallback

If consensus fails on certain values, parties have two choices: wait for consensus to return or handle the optimistic value as specified above. This could be a great option for some parties using data where quality can quickly change. A price feed for example might not come to consensus in times of api failures or exchange manipulations (e.g. feeds go down). In this case, it might be best to pause, but you could also go with optimistic values if your protocol needs a faster (albeit still slower than consensus) price. Note that in most cases, an uncertain value should NOT be pushed to your protocol. Tellor is unique in that rather than forcing reporters/ validators to sign off an api or price feed, the addition of their value is optional. This means that if a reporter is not certain it's a valid value, they will likely just sit out that round and the value will not reach consensus; something that should be seen as a good thing in cases where ambiguity still exists (e.g. two exchanges differ wildly on price).

Adding Security

Consumers of tellor data can add custom security in many ways. One way is to simply limit who on your chain can push prices/ state updates. By adding validation at this level, you could use either your own app-chains validators or stakers to push over the prices after validating them. This would be an excellent option for users with this ownership over their protocol. Another option for non-chains via the low latency model would be to add validation before a trusted party (e.g. the app's dao) pushes the root. It gives the trusted party an option to censor, but they would be unable to change what the price is, something that would work similarly to a multisig having pause authority/control over a protocol. You can also do OEV limits this way (relayer is a known party or even auction off the right for OEV each day/month)

Users can also add disputes on their end (similar to tellor disputing system, but can be handled by their governance). This would work well if coupled with custom staking requirements for reporters and could also be added as the "trusted" party that the tellor value must match.

Relayer

The relayer (pushing state roots from layer to user chains) can be anyone, however we will provide software for pushing values over and tipper scripts for setting up recurring feeds. Many keeper services also could fill this role (Keepr, Gelato, etc.).

Relay fallback

If there are no updates to a given light client bridge for >14 days (the staking/unstaking time on tellor chain), then the chain is considered stale, a situation where it needs a manual placement

[2] Depending on the type of data, this is a best practice for any system using an oracle

of a validator set. In this case, we will set the team's address as a fallback that can update this list. If parties do not want it to fallback to the team, they can simply update the contract once every 14 days or deploy a light client bridge without this fallback (or where it falls back to a different address).

Other usage methods
We know that some parties do not care as much about the decentralization of their bridge or already have existing solutions that they prefer. Layer data can be used via any of these solutions and we look forward to building and working with the teams to deliver the fastest and most secure solution for users. Some potential usage solutions include IBC, Hyperlane, Succinct, LayerZero, Chainlink, and many more.

In the future, it is likely that zero-knowledge bridges will be used to verify signatures, consensus, as well as inclusion of values. Tellor will be leaning on other teams currently specializing in cryptography research, but we fully expect that all bridges will be cheaper and faster using this method and should be operational within the next cycle.

Fork Choice

Upgrades and forks to layer will likely happen. Upgrades in the form of better data aggregation techniques, changes to the consensus protocol for some reason (e.g. cosmos sdk updates for faster blocktimes), or even changes to the tellor system to make it more secure. Hard forks on the other hand are security based and happen if the protocol is attacked, compromised, or broken in some way.

Soft upgrades
For upgrades, we can have the oracle itself report an upgrade. For each chain using the tellor protocol, we can have a mapping of chainID to a valid contract address for the verification of the data/protocol. In order to change it, a new light client contract with the updated code will be deployed on the EVM chain. Validators will then update the mapping, and propose a change on the consumer chain. Then after waiting 14 days (allows for users to exit if malicious), the address will be updated to the new validator contract.

Hard forks
For hard forks, you will also have a way to update this proxy address for verification of the consensus mechanism. The issue here is that time is of the essence. If the validator set is compromised or a bug is found, parties will want to very quickly switch off the oracle and upgrade. Unfortunately the switch cannot happen quickly, but freezing should be possible.[3]

[3] A library such as: https://github.com/RealityETH/subjectivocracy can be used as a dispute resolution mechanism for freezing and then voting on the results of the fork (all very costly to initiate to prevent censoring). Ultimately, you want this to be decided by the users on the chain itself and it might be a good choice to have a permissioned set (w/ a high cost still) or even a re-staking situation.

Tellor Token Bridge

The tellor token will be interchangeable between the current ERC20 TRB contract on Ethereum and the base token TRB on Layer. It will be secured by a two way bridge, operated by tellor layer itself via the trustless light client bridge. We will initially use a time-locked admin key to handle forks, but it will quickly move to just using the base layer bridge. In order to both incentivize the cosmos version and prevent attacks, withdrawals from Layer to Ethereum will be 28 days and can only be bridged if unstaked from being a validator on Layer. The bidirectional nature can be changed by forking Layer to not support it any longer and letting the deposit side lapse. Parties should forever have the ability to transfer tokens to the layer and it will likely take a long time for people to move over. It should be noted that the ultimate goal is that everyone moves away from the Ethereum token and to the Layer version.

Tellor will also need to limit how much can be bridged back to Ethereum. Upon launch of Layer, 300k new tokens will be minted and inflation / all rewards will start on Layer. All parties looking to validate on Layer will simply have to bridge their tellor tokens over.

Plan for legacy Tellor contracts

Tellor currently has users on Ethereum, Polygon, Gnosis, Arbitrum, and Optimism.

Tellor contracts are non-upgradeable, so users of the contracts can continue to use the contracts as they do currently. Users will need to continue to incentivize TRB token holders to stay on other chains posting data (tip), and hopefully will migrate over time as they upgrade their contracts or launch new systems.

For the ERC20 token contract, the minting of tokens will be given to the bridge contract (both for the team and oracle (time-based rewards) portions). The team will still maintain control of the team address (for voting purposes) and the oracle contract (for minting purposes and reading) will be the new tellor layer bridge. This means that the team mint contract will be an automated contract that gives minted tokens to the bridge, but gives the team the ability to vote through it. For the oracle contract, an oracle proxy will be placed over the normal bridge that allows users to read from the Layer user bridge (the light client contract).

The reason that minting is given to the bridge is to allow a two way bridge. Since there is minting on Layer, there would quickly become a mismatch between the two token supplies. This solution allows for greater flexibility around the support of legacy tellor contracts (we can bridge back to dispute/ vote) as well as gives unlimited time for exchanges/users in migrating to the new cosmos token.

Glossary

1. **Tellor Layer:** A standalone Layer 1 (L1) blockchain built using the Cosmos SDK, designed for consensus on subjective data using tendermint and an optimistic approach.
2. **Cosmos SDK**: A software development kit for building blockchains in the Cosmos ecosystem.
3. **Tendermint:** A Byzantine Fault Tolerant (BFT) consensus algorithm used by blockchains in the Cosmos network.
4. **Validator:** An entity in the Tellor Layer network that has locked a certain amount of TRB tokens to participate in block validation and data submission.
5. **Reporter:** An entity in the Tellor Layer network responsible for submitting values for data requests (queryIDs). Reporters can be staked validators or other participants, depending on the network's rules. They provide data, incentivized through tips, subject to validation and potential disputes.
6. **QueryID:** A unique identifier for a data request (e.g., BTC/USD price) on the Tellor Layer.
7. **TRB Token:** The native cryptocurrency of Tellor Layer, used for staking, tipping, and governance.
8. **Cycle List:** A governance-controlled list of queryIDs to maintain data freshness and ensure continuous reporting.
9. **Consensus Threshold:** The minimum amount of agreement required among validators for a value to be considered valid.
10. **Disputing**: A process in Tellor Layer where validators' submissions can be challenged, potentially leading to slashing of their stake.
11. **Slashing Event:** A penalty where a portion of a validator's stake is removed due to submission of incorrect data or other violations.
12. **Tipping:** A system where users can tip validators in TRB tokens for submitting data for specific queryIDs.
13. **Relayer:** An entity that transmits data from the Tellor Layer to other blockchains, facilitating cross-chain data accessibility and usage
14. **Consensus Usage:** A data verification method in Tellor Layer where values agreed upon by a supermajority of validators are instantly deemed reliable for real-time use.
15. **Optimistic Usage:** A method of using Tellor where submitted values are presumed correct if not disputed over a period of time, and/or X% of validators signed off.
16. **Light Client Bridge:** A mechanism for relaying Tellor Layer data to other blockchains.
17. **Validator Nonce:** A unique number used once by validators to prevent replay attacks.