The Flare network and FLR token

Flare Networks

December 31, 2022

Version 2.0

1 Introduction

Flare is a layer 1 Ethereum Virtual Machine (EVM) based blockchain which enables robust interoperability. The network is built with two open native interoperability protocols: the State Connector and the Flare Time Series Oracle (FTSO), which allow for on-chain decentralized acquisition of, respectively, blockchain and time series data. The State Connector enables decentralized consensus on the state of external blockchains, thus enabling on-chain acquisition of external blockchain data. The FTSO is a decentralized oracle for time series data, such as asset prices, data indices and more. These two open protocols allow for a wide array of applications that use off-chain data to be built. Crucially, these protocols are secured by the network itself.

This document proceeds as follows. Section 2 introduces the Flare network, the FLR token and its delegation mechanism. Next, the core protocols of the State Connector and FTSO are presented in section 3 and 4. Finally, consensus is presented in section 5, governance in section 6, the role of the Flare foundation in section 7 and finally the tokenomics in section 8.

2 The network and the token

2.1 The FLR token

Flare's native token, FLR, is required for native payments and spam control at the network level. Each FLR token has an associated and divisible vote which can be delegated in order to participate in the FTSO. Furthermore, as Flare uses proof of stake, both FLR and FLR's liquid staking token can be used as collateral within applications, as well as for governance purposes (see section 5).

Definition 2.1 *FLR is the native token of the Flare Network.*

FLR's technical purpose is to impose a cost upon transactions so as to disincentivize superfluous transactions i.e., network spamming. This is achieved via *transaction fees*. Flare Network uses the Ethereum Virtual Machine (EVM) [1], which defines a transaction's computational complexity in terms of units of gas.

2.2 FLR token delegation

FLR delegation, and more generally, programmable additions are achieved via *wrapped FLR*, WFLR. This is an ERC20 token supporting tracking and delegation of vote power and governance vote power to the FTSO and governance protocols respectively, which is in parity with FLR. Wrapped FLR are minted by depositing FLR to the token contract, and redeemed by withdrawing from the contract. In the following, WFLR and FLR are used interchangeably.

Definition 2.2 *Wrapped FLR, WFLR, is an ERC20 token issued one to one with FLR in order to achieve programmable functions.*

In order to maximize token utility, it is necessary that there be minimal competition between the use of FLR as collateral and that of delegation to the FTSO and governance. For instance, when FLR is used as collateral, it is locked in a smart contract so as to be available to compensate one or many counterparties of the relevant application. In this case, the collateral provider still potentially has a claim on all or part of it, which is enforced by the smart contract. In most applications, that claim will be the original collateral amount plus any additions less any losses and/or expenses. Usually the claim would correspond to the amount that the collateral provider could extract from the system if they were to unwind their participation in the application. This claim will likely be variable over time and is termed the *FLR claim*.

Definition 2.3 *The FLR claim is the amount of FLR tokens that are realizable from an application by an address if all participation in the application was unwound.*

If there was no way for the address to use the FLR claim amount to contribute to the FTSO (and potentially earn the FTSO reward), then the address's owner faces an undesirable opportunity cost in deploying that FLR as collateral in the application. A system of *delegation* is thus introduced to resolve this. Delegation allows an address to bestow all or a fraction of the votes associated with its FLR tokens upon another address for the purposes of both FTSO and governance participation *without* moving or transferring those tokens. Each FLR token holds votes that can be contributed to the FTSO and separate votes that can be contributed to governance. These votes may be delegated to different parties. The opportunity cost described above is then solved when any application that creates a FLR claim automates delegation of the claim amount to an address specified by the collateral provider.

Definition 2.4 *Delegation corresponds to an address on Flare assigning its own FLR token votes to another address on Flare.*

Token owners are free to delegate and undelegate their votes at any time. This leads naturally to the notion of an address's *vote power*, which is used both by the FTSO and governance.

Definition 2.5 *The number of non-delegated tokens held by an address plus any tokens delegated to that address give the address's* vote power*.*

To summarize, applications that use FLR as collateral and allow delegation of FLR tokens do not impact the FTSO or governance functions and usage of FLR as collateral does not undermine network safety. FLR holders can simultaneously earn the FTSO reward, return from any collateral, and continue to participate in governance.

3 The State Connector

3.1 Overview

The *State Connector* enables the *state* of an external blockchain to be captured in a decentralized manner on-chain. This is achieved by using two core protocols: the *request-commit-reveal* (RCR) and the *branching protocol*. The purpose of the RCR protocol is to gather user queries and proofs pertaining to blockchains, whereas the purpose of the branching protocol is for the network to accept or reject these.

3.2 The RCR protocol

The RCR protocol enables users to submit a request, termed an *attestation request*, to Flare in order to obtain information from another blockchain. It consists of three phases: request, commit and reveal. During the request phase, users can submit a query, which is a well-defined question. Acceptable requests are thus strictly defined by the system, in order to ensure they allow for no ambiguity i.e., well-formulated and deterministic. Furthermore, users must submit additional information alongside the query defining the range of blocks whereby the event to be queried occurred. For example, a user might submit a request to confirm that a transaction with specific parameters has not occurred within a given range of blocks.

Definition 3.1 *An* attestation request *is a well-defined query pertaining to an external blockchain.*

Next, *attestation providers* verify the query. For each type of query, a proof is unambiguously defined, termed an *attestation response*. The hash of this data is then taken, termed *attestation hash*.

Definition 3.2 Attestation providers *submit data proofs and participate in the RCR protocol.*

Definition 3.3 *An* attestation response*, or* attestation*, is the hash of data constituting the proof of a given request.*

Finally, in the reveal phase, attestation providers reveal their unhashed submissions. Each of the RCR protocol phases lasts 90 seconds, and thus a whole round lasts 4.5 minutes.

3.3 The branching protocol

An attestation provider is either part of Flare's *default set* or of a node's *local set*. The default set is a set of attestation providers initially set by the foundation, and subsequently determined by stake and key properties. In addition, each validator can choose a set of attestation providers, termed the validator's local set. These can be operated by anyone without any staking requirement. Note that the decentralization and security properties of the State Connector do not rely on the default set, as discussed in section 3.4.

Definition 3.4 Default set *attestation providers are attestation providers initially set by the foundation and subsequently determined by stake and key properties.*

Definition 3.5 Local set *attestation providers are a set of attestation providers chosen by a node, be it validator or observation.*

Following the reveal phase of the RCR protocol, the system checks whether more than 50% of the submitted attestations of the default set are identical. If so, this means that a majority of the attestation providers in the default set are in agreement. Next, each node checks whether more than 50% of the submitted attestations in their local set are identical and match the outcome of the default set.

3.4 Security

In the case of a malicious default set, two attack scenarios might occur: either they reach an incorrect majority agreement or they fail to reach an agreement.

In the case where an incorrect majority agreement is reached by the default set, an independent node acting as a local attestor would disagree. Its state database would then remain unchanged, i.e., as if the default set were unable to reach a majority vote on a State Connector round. Non-faulty core validators on a network leveraging the State Connector would also eventually remain in this unchanged state, which would then become the dominant network state; in the same manner as a new dominant chain history appearing in a proof-of-work network causes all nodes to automatically switch to that chain history [2].

For example, say that at block N, a voting round has just ended. At block $N+1$ votes are counted, and assume that a majority decision is reached by the default set, which is not accurate. For instance, a transaction that has not occurred is accepted as being confirmed. At this point, nodes running local attestors will disagree with this outcome, since it does not match the reality they observe. They will thus diverge as they disagree with the default result of the State Connector. At block $N+1$, all non-faulty independent nodes will have discarded the default outcome of the State Connector and thus once again the state of the network will be consistent with reality.

3.5 Optimisations

Various optimisations ensure that the system can handle a high volume of attestations. First, multiple rounds of the RCR rounds can be run in parallel. Second, in order to increase efficiency, an attestation provider can fulfill multiple requests submitted in a given request phase. This is achieved by using Merkle trees. Here, the attestation responses form the leaves from which a Merkle tree is computed. During the commit phase, the attestation provider sends the hash of the Merkle root, which is then revealed during the reveal phase. Note that attestation providers are thus required to answer all requests in a given phase, as otherwise this will give rise to potentially many different Merkle roots.

4 The FTSO

Decentralized oracles typically leverage the system participants in order to vote regarding the nature of a given proposition. For instance, the Schelling Coin protocol [3], uses a commit and reveal scheme, in order to gather votes from network participants. This results in a distribution over the votes, for which the median is computed and taken as the price estimate. The votes belonging within the interquartile range (IQR) are rewarded. Such schemes can be studied from a game theoretic perspective, whereby each participant has to make a choice in order to maximize their reward. For instance, the Schelling Coin scheme hinges on the idea that such a game has a Schelling point, i.e., a focal point [4].

Over recent years, the need to query information that is external to the chain has sparked the development of various approaches to oracles, such as TruthCoin consensus [5], Chainlink [6], Astraea [7] and Pyth network [8]. The core feature of an oracle should be a decentralized data feed exploiting the distributed nature of the underlying system while incentivizing players to be honest.

4.1 Overview

The FTSO is a decentralized protocol that aims to generate accurate estimates of off-chain time series data on the Flare Network. As in the Schelling protocol, at regular and prescribed

time intervals (a governance parameter), the oracle takes as input estimates from token holders and uses a weighted median algorithm to compute the output. Submissions falling in the interquartile range (IQR) are then rewarded. The system is designed to incentivize submission of accurate prices. At inception, the system will be providing prices in USD for XRP, ETH, BTC, and more.

The task of providing price estimates at regular time intervals in order to earn rewards is onerous for any given token holder. Thus, in practice, token holders will delegate their votes to *data providers*. They provide data estimates to the FTSO, and in return take a fee on rewards earned. In order to participate in the FTSO, a data provider must achieve a minimum threshold of votes delegated to it, while at the same time being capped. Furthermore, data providers are required to stake capital to ensure the system is Sybil resistant.

4.2 Computing the estimate

The FTSO yields an *oracle estimate* of a given price. These are computed by processing estimates submitted by token holders at regular time intervals, termed *price epochs*. During each price epoch, relevant token holders (as later defined) submit their data estimate, termed their *vote*, from which the oracle computes the *oracle estimate*. The votes are submitted using a commit and reveal scheme [9]. This scheme is designed to stop individuals from submitting prices based on other price proposals. Within the commit period, a hash of the data estimates (and some additional data, see section 4.4) is submitted, i.e., submissions are secret. After the commit period passes, submissions can no longer be changed. Next, in the reveal phase, the unhashed data is submitted, i.e., their prices are revealed. At this point, changes cannot be made, and prices become public record. Note that the commit and reveal rounds are overlapping, i.e., the reveal of one epoch takes place during the commit of the next round.

Definition 4.1 *A* price epoch *is a voting round, i.e., a time interval during which votes can be submitted.*

Definition 4.2 *The* vote v_t^i *at price epoch t is the submitted estimate of the i-th participating address.*

The FLR token holder votes are used to compute a weighted median, weighted by the address's vote power. The output is then the oracle estimate. More precisely, the weighted median m for an epoch is the value which minimizes

$$f(m) = \sum_{i=1}^{N_p} v_i \left| m - p_i \right|, \tag{1}$$

where N_p is the number of data providers, and the ith data provider with vote power v_i submits price p_i.

4.3 Reward mechanism

Token holders who submitted an estimate that belongs in the IQR, i.e., whose votes fall between the 25^{th} and 75^{th} percentile of the weighted estimate distribution, are rewarded. The precise amount of FLR tokens minted for rewards is a network governance parameter, which is termed the *price epoch reward*. Rewards can be claimed periodically, termed a *reward epoch*, and votes can be undelegated from an FTSO at any time.

Once per reward epoch, at a pseudo-random time, a snapshot of the delegation is taken in order to compute the number of votes delegated to each FTSO, i.e., the vote power. This is subsequently used for a series of price epochs.

Definition 4.3 *A* reward epoch *corresponds to a fixed number of voting rounds (price epochs), for which rewards can subsequently be claimed.*

Definition 4.4 *The* price epoch reward ρ *of an address captures the amount of rewards per unit time.*

4.4 Accessing pseudo-randomness on-chain

At each epoch, the compensation scheme requires access to a random number between 2^{128} and 2^{256}. Accessing randomness on-chain whilst not introducing an element of centralization is an active area of blockchain research and development [6, 10]. In order to choose the data feed to reward, the FTSO relies on the *on-chain pseudo-random number generator*, see Definition 4.5 and the output value must satisfy three requirements. First, it must be unpredictable. Second, it must be uniformly distributed between 2^{128} and 2^{256}. Third, it should be possible for external entities to verify the selected reward-FTSO after the fact (transparency).

The procedure is as follows. At each epoch t, data providers submit a random number between 1 and n inclusive alongside their price submissions. This number is denoted y_t^i, where $i = 1, \ldots, N_t$, for N_t data providers participating at epoch t.

Next, these are concatenated with the associated price submission, and the hash of this is taken, i.e., $\text{SHA256}(y_t^i || p_t^i)$, where $||$ denotes concatenation and p_t^i is the data estimate. These are then added and reduced modulo n. Finally, 1 is added to the result, in order to obtain an integer between 1 and n inclusive. This is the output of the *on-chain pseudo-random number generator*, summarized in the following Definition 4.5.

Definition 4.5 *In each price epoch t, the* on-chain pseudo-random number generator *takes as input a random integer y_t^i and price estimate p_t^i from each partaking account i in epoch t and computes n_t as follows:*

$$n_t = 1 + \sum_{i=1}^{N_t} SHA_{256}(y_t^i || v_t^i) \bmod n. \tag{2}$$

The on-chain pseudo-random number generator relies on FLR token holders to submit a random number. Note that it is not possible for the network to ascertain whether a given submitted value is indeed random or not. Instead, the system relies on the participants to be incentivized to submit a random number as requested. Since the random number is combined with the price estimate during the commit phase, the competing providers cannot guess the submitted price by bruteforce if the random number has good quality. If a competing provider can guess a submission, then their odds of winning increase. Thus, this acts as an incentive to submit good random numbers.

For FLR holders, the incentive is to obtain the reward and thus increase their FLR holdings. If they believe the FTSO chosen for reward is indeed random, then this incentivizes them to submit what they believe are true prices for *all* n data estimates, as this maximizes their chance of being within the IQR.

The on-chain pseudo-random number generator thus takes a combination of several pseudo-random sources submitted in a commit and reveal scheme. The construction does not reduce entropy, i.e., the output is as good as the quality of the pseudo-random numbers submitted. It is likely that the pseudo-random sources are system random number generators, and thus pseudo-random. If a true random number is submitted, then the output of the on-chain pseudo-random number generator will in turn be random. The on-chain pseudo-random number is then a seed of the on-chain randomness used for a given price epoch, as other values are derived from it.

4.5 Attack vectors

The FTSO builds on ideas introduced in the Schelling Coin [3], which is susceptible to the $P + \varepsilon$ attack [11]. The $P + \varepsilon$ [11] attack can be briefly characterized as follows. In a binary outcome process, a participant votes 1 or 0 according to what they believe the majority will vote, in order to earn a reward P. Each participant has one vote and there are no restrictions on who the participants are. The attacker commits to paying out $P + \varepsilon$ to those who vote 1 when the majority vote 0, where ε is some marginal amount. This skews the economic benefit for all participants to vote 1, regardless of the truth. Hence the majority of participants vote 1 and the attacker does not need to pay out. Pricing mechanisms such as Schelling Coin are susceptible to the $P + \varepsilon$ attack as these depend on how the majority of *participants* in the system will vote. In contrast, the FTSO weighs votes with token holdings. This means that the outcome of the vote is not dependent on the majority of participants, but instead on the majority of *stake*. Consequently, in order to corrupt the system, holders of a stake majority and delegation should vote a given way, which would be against their financial interest.

Furthermore, the FTSO uses a set of *trusted providers* as a backstop security mechanism. The trusted providers are initially picked by the Flare foundation (section 7). If the updated price deviates by more than 15% from the previous price, then the FTSO uses the price computed by the trusted providers. Thus, if an attacker attempts to manipulate the price, they may at best achieve a gain via arbitrage, which is thus limited.

Third, given the unknown token ownership distribution at any point in the future, it cannot therefore be automatically assumed, as it is in $P + \varepsilon$, that an attacker will successfully attack some future round of voting, the effects of which backpropagate to the current epoch.

For Flare, the $P+\varepsilon$ attack would require an attacker to convince a group of providers jointly holding a majority of token delegations to engage in the attack. Each provider participates in the FTSO in order to earn rewards over time. The ability of token holders to undelegate means that bad prices result in the loss of delegations, and thus potential earnings. Hence, an attacker must be able to fulfill an economic commitment to these providers superior to honest FTSO participation. Such a reward would only be credible if the attacker displayed an amount of capital committed to the attack that is the sum of the attack incentives as well as compensation for losses in value.

Thus, a successful attack requires all the following steps to be achieved. First, data providers holding a majority of delegations must agree to participate. Second, the potential payout is capped via arbitrage. Third, the change in price translates directly to the loss of value for the asset holders and must be compensated by the attacker.

5 Consensus

The network uses Avalanche consensus [12] along with proof of stake in order to ensure network security. Validators are required to stake a minimum amount in FLR.

6 Governance

Flare's token, FLR, is the governance mechanism for Flare [13]. Governance proposals on the network can originate either from the Flare Foundation or from the community via Songbird.

7 The Flare foundation

The Flare Foundation is a non-profit entity incorporated in the Netherlands. Its remit is to support the network traction and ongoing operations. This includes but is not limited to: grants, investments, partnerships, research and development, engineering, marketing, proposing and implementing governance, community and developer relations.

For this purpose the Flare Foundation receives 9,787,578,628 FLR at inception, of which 85% is escrowed and then distributed back to the Flare Foundation over 36 months. This is done in order to align the percentage of Flare Foundation tokens with the emission rate of FLR to the community, such that the Flare Foundation maintains a consistent ownership relative to the community over time.

8 FLR tokenomics and distribution

Flare's tokenomics and distribution schedule have been released in a separate document [14].

References

[1] Gavin Wood et al. "Ethereum: A secure decentralised generalised transaction ledger". In: ().

[2] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Decentralized Business Review* (2008), p. 21260.

[3] Vitalik Buterin. *SchellingCoin: A Minimal-Trust Universal Data Feed*. 2014. URL: https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/ (visited on 06/09/2020).

[4] Thomas C Schelling. *The strategy of conflict*. Harvard university press, 1980.

[5] Paul Sztorc. "Truthcoin". In: *peer-to-peer oracle system and prediction marketplace.* (2015).

[6] Steve Ellis, Ari Juels, and Sergey Nazarov. "Chainlink a decentralized oracle network". In: *Retrieved March* 11 (2017), p. 2018.

[7] John Adler et al. "Astraea: A decentralized blockchain oracle". In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE. 2018, pp. 1145–1152.

[8] Pyth data association. "Pyth network: a first-party financial oracle". In: (2022). URL: https://pyth.network/whitepaper.pdf.

[9] Oded Goldreich. *Foundations of cryptography: volume 1, basic tools*. Cambridge university press, 2007.

[10] Yossi Gilad et al. "Algorand: Scaling byzantine agreements for cryptocurrencies". In: *Proceedings of the 26th Symposium on Operating Systems Principles*. 2017, pp. 51–68.

[11] Vitalik Buterin. *The P + epsilon Attack*. 2015. URL: https://blog.ethereum.org/2015/01/28/p-epsilon-attack/ (visited on 06/09/2020).

[12] Team Rocket. *Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies*. 2018.

[13] Flare Network. *Flare governance and the role of Songbird*. 2022. URL: https://flare.xyz/governance/ (visited on 10/27/2022).

[14] Flare Network. *Tokenomics detail*. 2022. URL: https://flare.network/tokenomics/ (visited on 12/22/2022).