TORTUGA LIQUID STAKING PROTOCOL WHITEPAPER

Abstract. Tortuga is an open-source protocol for validators on Aptos to receive delegations from multiple APT addresses, which the Aptos core framework does not support at launch.

The protocol also aims at (1) maximizing total staked APT coins by creating a low entry for participation in APT staking for end users, thus increasing the security of the Aptos blockchain, and (2) creating a more equitable distribution of voting power by letting end users and permissionless oracles determine the stake distribution among validators, thus driving up decentralization.

1. Introduction

Aptos is a new proof-of-stake blockchain running on *Move*, a parallelizable and type-safe programming language that provides strict safety guarantees; see [1]. The chain supports high throughput (over 100k TPS) with low transaction fees.

The validator nodes, henceforth *validators* securing Aptos must perform at the highest level for it to achieve these goals. They run on expensive hardware and are constantly monitored by operators. Aptos incentivizes them to provide this quality of service by issuing APT proportional to the APT staked with them.

Aptos launched with a limited set of validators. The current minimum stake of 1M APT constrains the number of operators, though it is expected to decrease as the network stabilizes.

Natively, validators only represent staked APT from a single address. This address can either be controlled by the operator or another entity, such as the Aptos foundation. Validators cannot accept outside delegations. This limits the commissions that operators can earn and invest in improving hardware and operations.

Addresses holding less than 1M APT are not able to stake due to this constraint.

Validators who use the Tortuga protocol can accept delegations. Addresses that previously could not stake, now can, by delegating to these validators without requiring any overseeing entities[1].

Delegators can choose to stake as little as 0.01 APT with either (1) a specific Tortuga Protocol validator based on their preference, or (2) allow the protocol to determine the best stake distribution amongst its validators based on node performance, commission, and overall system decentralization. In the latter case, delegators receive tAPT to denote their participation in the protocol.

2. Flow of funds in Tortuga

The Tortuga protocol accepts APT and returns tAPT. tAPT may be exchanged with the protocol for APT with a delay of up to one unlock period, which is set by the Aptos chain. This is described in §4.4. The Tortuga protocol puts the APT in *reserve*. The protocol may stake or unstake the *reserve* with its *associated validators*, that is, the validators interfacing with the Tortuga open-source code. The protocol is in *deficit* if the APT in the *reserve* is less than the APT required to service tAPT swaps. This can happen because APT is temporarily locked by Aptos when it is staked. In this case, the contract can be called permissionlessly to unstake APT from associated validators. If the protocol is *not* in deficit, then *total stakable amount* that can be staked with associated validators is nonzero and anyone can increase stake in associated validators.

Date: November 2, 2022.

[1]At times, a permissionless bot is needed in order to maintain the state of the underlying contract. An on-chain oracle is a typical example of this.

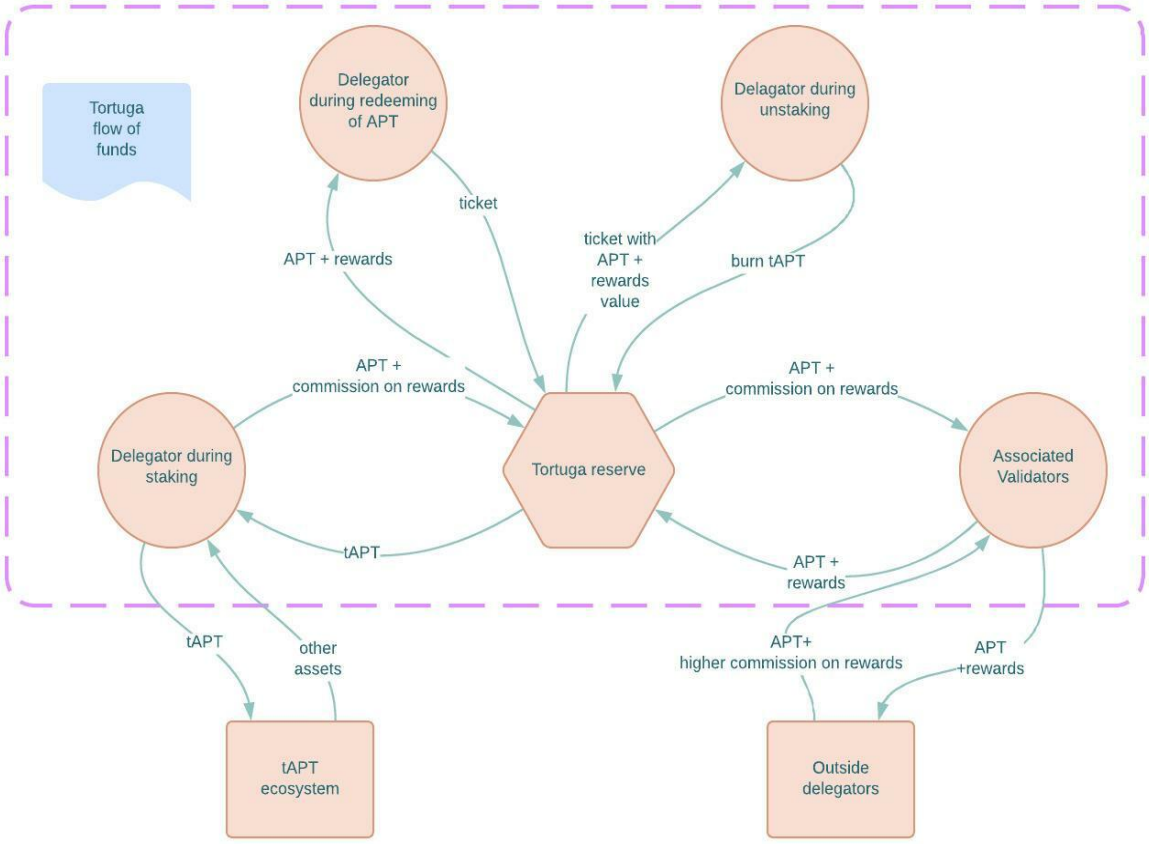

FIGURE 1. Flow of funds in Tortuga. Rewards are staking yield generated by the Aptos blockchain. The validators charge commissions (about 0% – 12%) on the staking yield they generate for individual delegators as well as Tortuga protocol.

The protocol uses *withdraw signatures* to unstake APT from the validators that unlock next to exit *deficit* quickly.

A withdraw signature stores the address and the unlocking time of an associated validator. This signature remains valid as long as the associated validator remains associated and its unlocking time does not change. Thus if the validator's auto-lockup is renewed by the Aptos staking mechanism, the withdraw signature becomes invalid. In case of a deficit, anyone can request to unlock from any associated validator if there is no valid withdraw signature. If there is a valid withdraw signature, then the unlock request can only be placed with an associated validator whose unlocking time is on or before the unlocking time stored in the withdraw signature. In particular, in case of a deficit, anyone can always place an unlock request with the earliest unlocking associated validator, no matter what the withdraw signature is. This ensures that APT is never locked in Tortuga even if the withdraw signature stops being updated.

3. Participating in Tortuga as a validator

3.1. **Validator signup.** The Tortuga protocol supports permissionless validator registration. Any validator will be able to become associated with Tortuga, as long as they are active and maintain the minimum stake as defined by the Aptos staking mechanism. At registration, a validator can specify the share of newly generated APT attributable to them for delegations from the protocol and direct delegations separately. This value is commonly called *commission* in proof-of-stake

systems. The protocol requires that the delegations via the protocol are not subject to higher commission compared to direct delegations. Since Aptos blockchain does not provide a native delegation service, signing up with Tortuga is an attractive option for validators as it allows them to accept outside delegations.

An associated validator can designate any address to receive their share of newly generated APT. This address can be rotated at any time.

3.2. **Getting stake and earning rewards.** A new validator that gets associated with Tortuga starts with a score of zero, which increases overtime. Even with a score of zero, Tortuga's target delegation for a new validator is positive, although very small. At any point in time, if the protocol is not in a deficit (*i.e.*, there is more unstaked APT in the protocol than are required to disburse any pending unstake claims from the delegators), the validator can permissionlessly add any amount up to their target delegation to themselves.

The Tortuga protocol provides a permissionless endpoint to the validators to charge commission as often as they want. This lets them compound their commission. For example, if a delegator delegates 100,000 APT with a validator who charges 50% commission and the rewards are generated at the rate of 2% per month, then the operator can either earn 2020 APT if they charge commission at the end of two months, or they can earn $1000(1.02) + 1010 = 2030$ APT at the end of two months if they charge commission every month. If anyone using the protocol interacts with an associated validator via a delegation, unlocking, or withdrawal, they trigger commission pay to that validator.

3.3. **Scoring and determination of target delegations for associated validators.** The Tortuga protocol uses an open-source on-chain oracle that tracks *time weighted average effective reward rate*, henceforth *twar*, for each of its associated validators. Figure 2, depicts the twar and corresponding observations of a mainnet associated validator at the time of writing.

A validator's twar is calculated after accounting for their on-chain performance degradations and after paying commission to them. Thus, twar is correlated with validator's on-chain performance which ultimately results in a more equitable distribution of stake among the validators.

The protocol's *score* for a validator is given by

```
score = ramp_up_multiplier * cliff(commission_change_multiplier * validator_twar),
```

where *cliff* is a rapidly decreasing function. The protocol's target delegation is directly proportional to its score. The following examples explain the behavior of the score function under normal circumstances:

(1) A 100% performing validator which charges Tortuga 0% commission has a score of 10^{10}.
(2) A 96% performing validator which charges Tortuga 0% commission has a score of 10^{8}.
(3) A 100% performing validator which charges Tortuga 4% commission has a score of 10^{8}.
(4) A 92% performing validator which charges Tortuga 0% commission has a score of 10^{6}.
(5) A 96% performing validator which charges Tortuga 4% commission has a score of 10^{6}.

Thus if a validator with stats as in (1) has a target delegation of 10M APT, then the validator with stats as in (2) has a target delegation of 100k APT. This steepness of the cliff function pushes validators to perform optimally and offer lower commission to the protocol as long as the protocol is able to provide them with enough stake. Under some special circumstances, one or both of the *ramp up multiplier* and *commission change multiplier* may kick in.

The ramp up multiplier makes sure that the validator score ramps up slowly over time after registering with the protocol. This prevents a malicious validator operator from repeatedly performing badly, offboarding, and then quickly registering again with a new address in order to delete the scoring history.

```
AssociatedValidator {
    "last_update_timestamp": 1667327533,
    "last_reconfiguration_timestamp": 1667324146,
    "balance_at_last_update_unreserved": 54948276266277,
    "balance_at_last_update_reserved": 0,
    "time_averaged_effective_reward_rate": 142639925898,
    "observations": CircularBuffer {
     "buffer": [
       Observation {
         "timestamp": 1667194544,
         "effective_reward_rate": 143677332936
       },
       Observation {
         "timestamp": 1667230544,
         "effective_reward_rate": 143762496586
       },
       Observation {
         "timestamp": 1667266545,
         "effective_reward_rate": 143808418756
       },
       Observation {
         "timestamp": 1667324146,
         "effective_reward_rate": 143834547958
       },
       Observation {
         "timestamp": 1666884938,
         "effective_reward_rate": 143672062036
       },
       Observation {
         "timestamp": 1666964139,
         "effective_reward_rate": 143624890329
       },
       Observation {
         "timestamp": 1667064941,
         "effective_reward_rate": 143730793016
       },
       Observation {
         "timestamp": 1667079342,
         "effective_reward_rate": 113554128763
       Observation {
         "timestamp": 1667151343,
         "effective_reward_rate": 141941998267
       },
       Observation {
         "timestamp": 1667180143,
         "effective_reward_rate": 143791575437
       }
     ],
     "last_index": 3
    },
    "observation_begin_timestamp": 1666805737,
    "score": 14510,
    "permissioned_score": 10000000000,
    "ramp_up_start_at": 1666213653
  }
```


FIGURE 2. Time averaged effective reward rate and score for a mainnet associated validator that charges 10% commission to the Tortuga protocol.

A validator can request to change commission at any point. If the new commission is higher, the change takes effect after the Aptos Core Framework unlock period (currently 30 days) to ensure that the protocol and outside delegators have sufficient time to make changes to delegations. The commission change multiplier gets activated as soon as a commission change request is placed by

an associated validator, and the target delegation for that validator gets adjusted before the next lockup cycle starts.

3.4. **Security of validator's funds.** The protocol's design is such that a new delegator's total worth with a validator decreases for the duration of the current epoch (roughly 2 hours), only to increase again beyond the delegated amount after a span of one epoch. This prevents a potential attack in which a delegator delegates funds to a validator moments before the lockup expires and takes a part of the issued APT.

The protocol also provides a safe way to retrieve minimum stake that validators have to maintain in order to be associated with Tortuga. They can simply opt to leave and retrieve their self-stake at the expiration of the current lockup.

4. Participating in Tortuga as a delegator

4.1. **Getting tAPT.** The *stake* function accepts APT and returns tAPT at the current tAPT-to-APT conversion rate. This rate is based on the ratio of APT in the protocol to tAPT in existence. It increases as new APT is generated through staking and tAPT is burned for community rebates; see §4.3. Note that the conversion rate can change without new APT being generated, for example, initial delegation loss as explained in §3.4 can temporarily decrease the conversion rate.

There is no protocol fee on delegating, and the delegated amount can be as little as 0.01 APT.

4.2. **Redeeming APT and generated staking yield.** The *unstake* call accepts tAPT. A fixed small percentage is burned in a community rebate; see §4.3. The amount of remaining tAPT is converted to APT at the current conversion rate and returns a ticket storing that value. The remaining tAPT is then burned. That amount of APT can be claimed once it is available in the reserve. There is no protocol fee for claiming. As explained in §2, if the protocol is in deficit, APT cannot be claimed immediately. In that case, APT can be unlocked permissionlessly from the earliest unlocking validators (as dictated by withdraw signatures) until the protocol is no longer in deficit.

The protocol avoids dust below a threshold of 0.01 APT. As an example, if the user's account holds 10.23343438 tAPT and user calls to unstake 10.23 tAPT, the protocol automatically clears the dust and issues them a ticket for the whole of 10.2334343 tAPT. Both tAPT and APT have 8 decimal point precision.

4.3. **Community rebate.** A small fixed percentage of the tAPT passed to *unstake* is burned before the APT value of the ticket is calculated. This increases the conversion rate.

4.4. **Security of delegators' funds.** The protocol employs a first-in-first-out strategy for tickets. A ticket stores the amount of redeemable APT, and an *id* equal to the sum of the redeemable amounts of all issued tickets before the given ticket. The protocol also stores the sum of all ticket amounts that have already been redeemed. A given ticket can be redeemed when:

```
ticket_id + ticket_amount <= reserve_balance + total_of_redeemed_tickets.
```

This ensures that if ticket A is issued before ticket B, then ticket A is eligible to be claimed before ticket B. If ticket B is redeemed before ticket A, then ticket A can always be redeemed, no matter what the deficit is.

To prevent overflows over any meaningful timescale, all ticket related values in the protocol, including the *id* of a ticket, are stored in the `u128` format, which is the largest integer format natively supported by the Aptos blockchain.

5. Conclusion

The Tortuga protocol allows validators to accept delegations, both directly and through the protocol's routing algorithm. The protocol can accept APT and return tAPT based on a transparent algorithm.

Tortuga reduces friction for staking, which should increase staking participation and Aptos network security.

References

[1] The Aptos Blockchain: A Safe, Scalable, and Upgradeable Web3 Infrastructure, available at https://aptos.dev/aptos-white-paper/aptos-white-paper-index/.