9 Delegation

We briefly describe the motivation and context for delegation. The full context is contained in [SL-D1].

Stake is said to be *active* in the blockchain protocol when it is eligible for participation in the leader election. In order for stake to become active, the associated verification stake credential must be registered and its staking rights must be delegated to an active stake pool. Individuals who wish to participate in the protocol can register themselves as a stake pool.

Stake credentials are registered (or deregistered) through the use of registration (or deregistration) certificates. Registered stake credentials are delegated through the use of delegation certificates. Finally, stake pools are registered (or retired) through the use of registration (or retirement) certificates.

Stake pool retirement is handled a bit differently than stake deregistration. Stake credentials are considered inactive as soon as a deregistration certificate is applied to the ledger state. Stake pool retirement certificates, however, specify the epoch in which it will retire.

Delegation requires the following to be tracked by the ledger state: the registered stake credentials, the delegation map from registered stake credentials to stake pools, pointers associated with stake credentials, the registered stake pools and upcoming stake pool retirements. Additionally, the blockchain protocol rewards eligible stake and so we must also include a mapping from active stake credentials to rewards.

Finally, there are two types of delegation certificates available only to the genesis keys. The genesis keys will still be used for update proposals at the beginning of the Shelley era, and so there must be a way to maintain the delegation of these keys to their cold keys. This mapping is also maintained by the delegation state. There is also a mechanism to transfer rewards directly from either the reserves pot or the treasury pot to a reward address. While technically everybody can post such a certificate, the transaction that contains it must be signed by Quorum-many genesis key delegates.

9.1 Delegation Definitions

In Figure 21 we give the delegation primitives. Here we introduce the following primitive datatypes used in delegation:

- $\mathsf{DCert}_{\mathsf{regkey}}$: a stake credential registration certificate.
- $\mathsf{DCert}_{\mathsf{deregkey}}$: a stake credential de-registration certificate.
- $\mathsf{DCert}_{\mathsf{delegate}}$: a stake credential delegation certificate.
- $\mathsf{DCert}_{\mathsf{regpool}}$: a stake pool registration certificate.
- $\mathsf{DCert}_{\mathsf{retirepool}}$: a stake pool retirement certificate.
- $\mathsf{DCert}_{\mathsf{genesis}}$: a genesis key delegation certificate.
- $\mathsf{DCert}_{\mathsf{mir}}$: a move instantaneous rewards certificate.
- DCert: any one of of the seven certificate types above.

The following derived types are introduced:

- $\mathsf{PoolParam}$ represents the parameters found in a stake pool registration certificate that must be tracked:
  - the pool owners.

  - the pool cost.
  - the pool margin.
  - the pool pledge.
  - the pool reward account.
  - the hash of the VRF verification key.
  - the pool relays.
  - optional pool metadata (a url and a hash).

  The idea of pool owners is explained in Section 4.4.4 of [SL-D1]. The pool cost and margin indicate how much more of the rewards pool leaders get than the members. The pool pledge is explained in Section 5.1 of [SL-D1]. The pool reward account is where all pool rewards go. The pool relays and metadata url are explained in Sections 3.4.4 and 4.2 of [SL-D1].

Accessor functions for certificates and pool parameters are also defined, but only the $\mathsf{cwitness}$ accessor function needs explanation. It does the following:

- For a $\mathsf{DCert}_{\mathsf{regkey}}$ certificate, $\mathsf{cwitness}$ is not defined as stake key registrations do not require a witness.
- For a $\mathsf{DCert}_{\mathsf{deregkey}}$ certificate, $\mathsf{cwitness}$ returns the hashkey of the key being de-registered.
- For a $\mathsf{DCert}_{\mathsf{delegate}}$ certificate, $\mathsf{cwitness}$ returns the hashkey of the key that is delegating (and not the key to which the stake in being delegated to).
- For a $\mathsf{DCert}_{\mathsf{regpool}}$ certificate, $\mathsf{cwitness}$ returns the hashkey of the key of the pool operator.
- For a $\mathsf{DCert}_{\mathsf{retirepool}}$ certificate, $\mathsf{cwitness}$ returns the hashkey of the key of the pool operator.
- For a $\mathsf{DCert}_{\mathsf{genesis}}$ certificate, $\mathsf{cwitness}$ returns the hashkey of the genesis key.
- For a $\mathsf{DCert}_{\mathsf{mir}}$ certificate, $\mathsf{cwitness}$ is not defined as there is no single core node or genesis key that posts the certificate.

Abstract types

$$\begin{array}{rcll} \mathit{url} & \in & \mathsf{Url} & \text{a url} \\ \mathit{mp} & \in & \mathsf{MIRPot} & \text{either } \mathsf{ReservesMIR} \text{ or } \mathsf{TreasuryMIR} \end{array}$$

Delegation Certificate types

$$\begin{aligned} \mathsf{DCert} = {} & \mathsf{DCert}_{\mathsf{regkey}} \uplus \mathsf{DCert}_{\mathsf{deregkey}} \uplus \mathsf{DCert}_{\mathsf{delegate}} \\ & \uplus \mathsf{DCert}_{\mathsf{regpool}} \uplus \mathsf{DCert}_{\mathsf{retirepool}} \uplus \mathsf{DCert}_{\mathsf{genesis}} \\ & \uplus \mathsf{DCert}_{\mathsf{mir}} \end{aligned}$$

Derived types

$$\begin{array}{rcll} \mathsf{PoolMD} & = & \mathsf{Url} \times \mathsf{PoolMDHash} & \text{stake pool metadata} \\ \mathsf{PoolParam} & = & \mathbb{P}\,\mathsf{KeyHash} \times \mathsf{Coin} \times [0,\,1] \times \mathsf{Coin} & \text{stake pool parameters} \\ & & \quad \times \mathsf{Addr}_{\mathsf{rwd}} \times \mathsf{KeyHash}_{vrf} & \\ & & \quad \mathsf{Url}^{*} \times \mathsf{PoolMD}^{?} & \end{array}$$

Certificate Accessor functions

$$\begin{array}{rcll} \mathsf{cwitness} & \in & \mathsf{DCert} \setminus (\mathsf{DCert}_{\mathsf{regkey}} \cup \mathsf{DCert}_{\mathsf{mir}}) \to \mathsf{Credential} & \text{certificate witness} \\ \mathsf{regCred} & \in & \mathsf{DCert}_{\mathsf{regkey}} \to \mathsf{Credential} & \text{registered credential} \\ \mathsf{dpool} & \in & \mathsf{DCert}_{\mathsf{delegate}} \to \mathsf{KeyHash} & \text{pool being delegated to} \\ \mathsf{poolParam} & \in & \mathsf{DCert}_{\mathsf{regpool}} \to \mathsf{PoolParam} & \text{stake pool} \\ \mathsf{retire} & \in & \mathsf{DCert}_{\mathsf{retirepool}} \to \mathsf{Epoch} & \text{epoch of pool retirement} \\ \mathsf{genesisDeleg} & \in & \mathsf{DCert}_{\mathsf{genesis}} \to (\mathsf{KeyHash}_{\mathsf{G}},\, \mathsf{KeyHash},\, \mathsf{KeyHash}_{vrf}) & \text{genesis delegation} \\ \mathsf{credCoinMap} & \in & \mathsf{DCert}_{\mathsf{mir}} \to (\mathsf{Credential}_{stake} \mapsto \mathsf{Coin}) & \text{moved inst. rewards} \\ \mathsf{mirPot} & \in & \mathsf{DCert}_{\mathsf{mir}} \to \mathsf{MIRPot} & \text{pot for inst. rewards} \end{array}$$

Pool Parameter Accessor functions

$$\begin{array}{rcll} \mathsf{poolOwners} & \in & \mathsf{PoolParam} \to \mathbb{P}\,\mathsf{KeyHash} & \text{stake pool owners} \\ \mathsf{poolCost} & \in & \mathsf{PoolParam} \to \mathsf{Coin} & \text{stake pool cost} \\ \mathsf{poolMargin} & \in & \mathsf{PoolParam} \to [0,\,1] & \text{stake pool margin} \\ \mathsf{poolPledge} & \in & \mathsf{PoolParam} \to \mathsf{Coin} & \text{stake pool pledge} \\ \mathsf{poolRAcnt} & \in & \mathsf{PoolParam} \to \mathsf{Addr}_{\mathsf{rwd}} & \text{stake pool reward account} \\ \mathsf{poolVRF} & \in & \mathsf{PoolParam} \to \mathsf{KeyHash}_{vrf} & \text{stake pool VRF key hash} \end{array}$$

Figure 21: Delegation Definitions

9.2 Delegation Transitions

In Figure 22 we give the delegation and stake pool state transition types. We define two separate parts of the ledger state.

- DState keeps track of the delegation state, consisting of:
  - *rewards* stores the rewards accumulated by stake credentials. These are represented by a finite map from reward addresses to the accumulated rewards.
  - *delegations* stores the delegation relation, mapping stake credentials to the pool to which is delegates.
  - *ptrs* maps stake credentials to the position of the registration certificate in the blockchain. This is needed to lookup the stake hashkey of a pointer address.
  - *fGenDelegs* are the future genesis keys delegations. This variable is needed because genesis keys can only update their delegation with a delay of StabilityWindow slots after submitting the certificate (this is necessary for header validation, see Section 12)
  - *genDelegs* maps genesis key hashes to hashes of the cold key delegates.
  - i_{rwd} stores two maps of stake credentials to Coin, which is used for moving instantaneous rewards at the epoch boundary. One map corresponds to rewards taken from the reserves, and the other corresponds to rewards taken from the treasury.
- PState keeps track of the stake pool information:
  - *poolParams* tracks the parameters associated with each stake pool, such as their costs and margin.
  - When changes are made to the pool parameters late in an epoch, they are staged in *fPoolParams*. These parameters will be updated by another transition (namely EPOCH) when the next epoch starts.
  - *retiring* tracks stake pool retirements, using a map from hashkeys to the epoch in which it will retire.

The operational certificates counters *cs* in the stake pool state are a tool to ensure that blocks containing outdated certificates are rejected. These certificates are part of the block header. For a discussion of why this additional mechanism is needed, see the document [SL-D1], and for the relevant rules, see Section 12.8.

The environment for the state transition for DState contains the current slot number, the index for the current certificate pointer, and the account state. The environment for the state transition for PState contains the current slot number and the protocol parameters.

Delegation Types

$$
\begin{aligned}
\mathit{stakeCred} &\in \mathsf{Credential}_{\mathit{stake}} &&= (\mathsf{KeyHash}_{\mathit{stake}} \uplus \mathsf{ScriptHash})\\
\mathit{fGenDelegs} &\in \mathsf{FutGenesisDelegation} &&= (\mathsf{Slot} \times \mathsf{KeyHash}_{\mathsf{G}}) \mapsto (\mathsf{KeyHash} \times \mathsf{KeyHash}_{\mathit{vrf}})\\
\mathit{ir} &\in \mathsf{InstantaneousRewards} &&= (\mathsf{Credential}_{\mathit{stake}} \mapsto \mathsf{Coin})\\
& && \quad \times (\mathsf{Credential}_{\mathit{stake}} \mapsto \mathsf{Coin})
\end{aligned}
$$

Delegation States

$$
\mathsf{DState} = \left(\begin{array}{rlr}
\mathit{rewards} & \in \mathsf{Credential}_{\mathit{stake}} \mapsto \mathsf{Coin} & \text{rewards}\\
\mathit{delegations} & \in \mathsf{Credential}_{\mathit{stake}} \mapsto \mathsf{KeyHash}_{\mathit{pool}} & \text{delegations}\\
\mathit{ptrs} & \in \mathsf{Ptr} \mapsto \mathsf{Credential}_{\mathit{stake}} & \text{pointer to stake credential}\\
\mathit{fGenDelegs} & \in \mathsf{FutGenesisDelegation} & \text{future genesis key delegations}\\
\mathit{genDelegs} & \in \mathsf{GenesisDelegation} & \text{genesis key delegations}\\
i_{\mathit{rwd}} & \in \mathsf{InstantaneousRewards} & \text{instantaneous rewards}
\end{array}\right)
$$

$$
\mathsf{PState} = \left(\begin{array}{rlr}
\mathit{poolParams} & \in \mathsf{KeyHash}_{\mathit{pool}} \mapsto \mathsf{PoolParam} & \text{registered pools to pool parameters}\\
\mathit{fPoolParams} & \in \mathsf{KeyHash}_{\mathit{pool}} \mapsto \mathsf{PoolParam} & \text{future pool parameters}\\
\mathit{retiring} & \in \mathsf{KeyHash}_{\mathit{pool}} \mapsto \mathsf{Epoch} & \text{retiring stake pools}
\end{array}\right)
$$

Delegation Environment

$$
\mathsf{DEnv} = \left(\begin{array}{rlr}
\mathit{slot} & \in \mathsf{Slot} & \text{slot}\\
\mathit{ptr} & \in \mathsf{Ptr} & \text{certificate pointer}\\
\mathit{acnt} & \in \mathsf{Acnt} & \text{accounting state}
\end{array}\right)
$$

Pool Environment

$$
\mathsf{PEnv} = \left(\begin{array}{rlr}
\mathit{slot} & \in \mathsf{Slot} & \text{slot}\\
\mathit{pp} & \in \mathsf{PParams} & \text{protocol parameters}
\end{array}\right)
$$

Delegation Transitions

$$
\_ \vdash \_ \xrightarrow[\mathsf{DELEG}]{\_} \_ \in \mathbb{P}\,(\mathsf{DEnv} \times \mathsf{DState} \times \mathsf{DCert} \times \mathsf{DState})
$$

$$
\_ \vdash \_ \xrightarrow[\mathsf{POOL}]{\_} \_ \in \mathbb{P}\,(\mathsf{PEnv} \times \mathsf{PState} \times \mathsf{DCert} \times \mathsf{PState})
$$

Figure 22: Delegation Transitions