Tezos economic protocol

Eugen Zălinescu

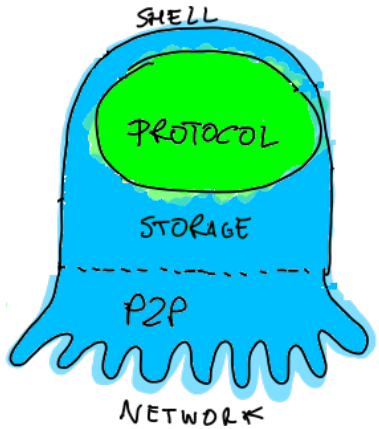
Nomadic Labs training sessions

11-02-2020

Tezos node

Tezos node = the **economic protocol** plugged in a generic **shell**

- economic protocol = Tezos ruleset
  - what is a valid block and a valid operation
  - how is the ledger updated
  - one of the rules: rules can be changed
- shell = the network and storage layer
  - The network layer implements a P2P gossip protocol.
  - The storage layer stores seen blocks.



What's in the economic protocol?

- Abstractly
  - Proof-of-stake system
  - Governance
  - Transaction layer (incl. Michelson, the smart-contract language)
- Concretely
  - blocks and operations
  - the ledger: money, accounts, smart contracts, ...
  - RPCs, constants, ...

Outline

- **Liquid Proof-of-Stake and Consensus**
- Protocol amendment

Liquid Proof-of-stake

- Proof-of-stake system:
  - Blocks can only be produced by token owners.
  - Rights to produce blocks are given randomly in proportion to stake.
- Delegation:
  - Token owners can delegate their tokens.
  - Delegates cannot spend their delegated tokens.
  - Delegates can be chosen or revoked at any time.
- Role of a delegate is to take part in:
  - the consensus algorithm
  - the voting process
- There are currently over 400 delegates.
  - 81% of monetary mass takes part in consensus
  - 79% of it is delegated, not owned

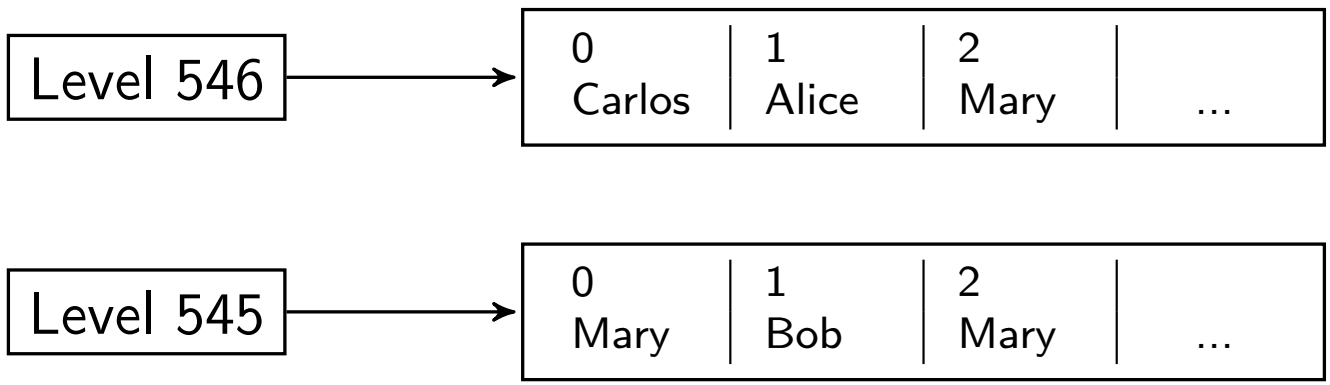
Emmy$^+$

- Emmy: state of the art PoS algorithm in 2014
- Emmy$^+$: improved version adopted in Babylon (Oct.2019)

- We'll proceed incrementally:
  - building blocks
  - simplified version
  - endorsements
  - incentives

Baking rights

- For each level, an infinite list of bakers is drawn at random.
  - implemented through a follow-the-coin procedure
    - track owner of each roll; 1 roll = 8000 ꜩ
    - choose a roll randomly
  - baker/block *priority* = index in the list



...

Seeds and roll snapshots

Seeds

- The same baking rights must be computed by every node.
  - Solution: use a PRNG with a random seed
  - Seed computation based on a commit & reveal scheme.

Seeds and roll snapshots

Seeds

- The same baking rights must be computed by every node.
  - Solution: use a PRNG with a random seed
  - Seed computation based on a commit & reveal scheme.

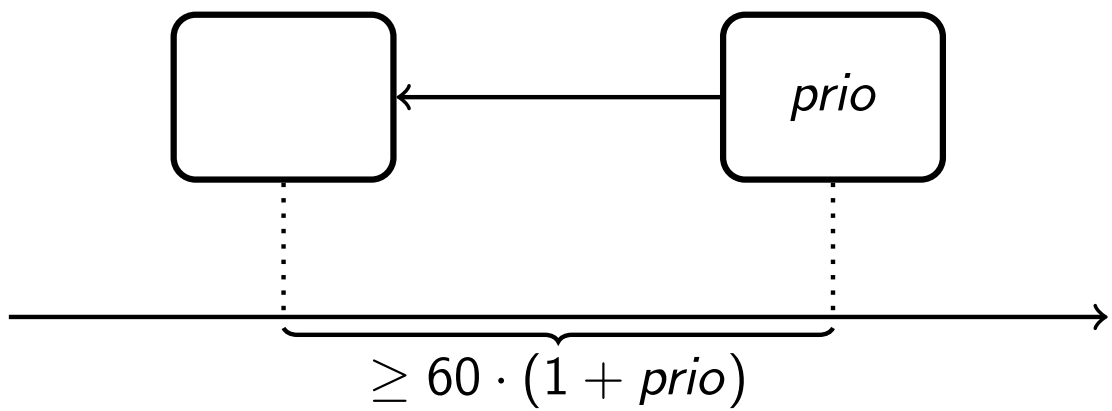
Roll snapshots

- Token ownership varies over time.
- To prevent manipulation:
  - take regular snapshots of the token distribution
  - choose snapshot at random.

Simple Emmy$^+$

- Recall: Tezos node = shell + economic protocol
  - The shell maintains the main chain: the valid one with the highest *score*.
  - The protocol defines the score and the validity rules.

- Simple Emmy$^+$ rules:
  - score = chain length
  - validity = minimal time between blocks:



Simple Emmy$^+$: assumptions and properties

- Assumptions
  - Nodes have access to loosly synchronized clocks.
  - The network is synchronous: message delays less than 30sec.
- These should ensure *probabilistic finality*
  - in the short run there can be forks:
    - assume the attacker has priority 0 for levels $\ell + 1, \ldots, \ell + k$
      $\rightarrow$ this happens with probability s^{-k}
    - she proposes the chain $B_{\ell+1} \ldots B_{\ell+k}$ to some nodes
    - she proposes the chain $B'_{\ell+1} \ldots B'_{\ell+k}$ to some other nodes
  - in the long run, the honest chain is the faster chain

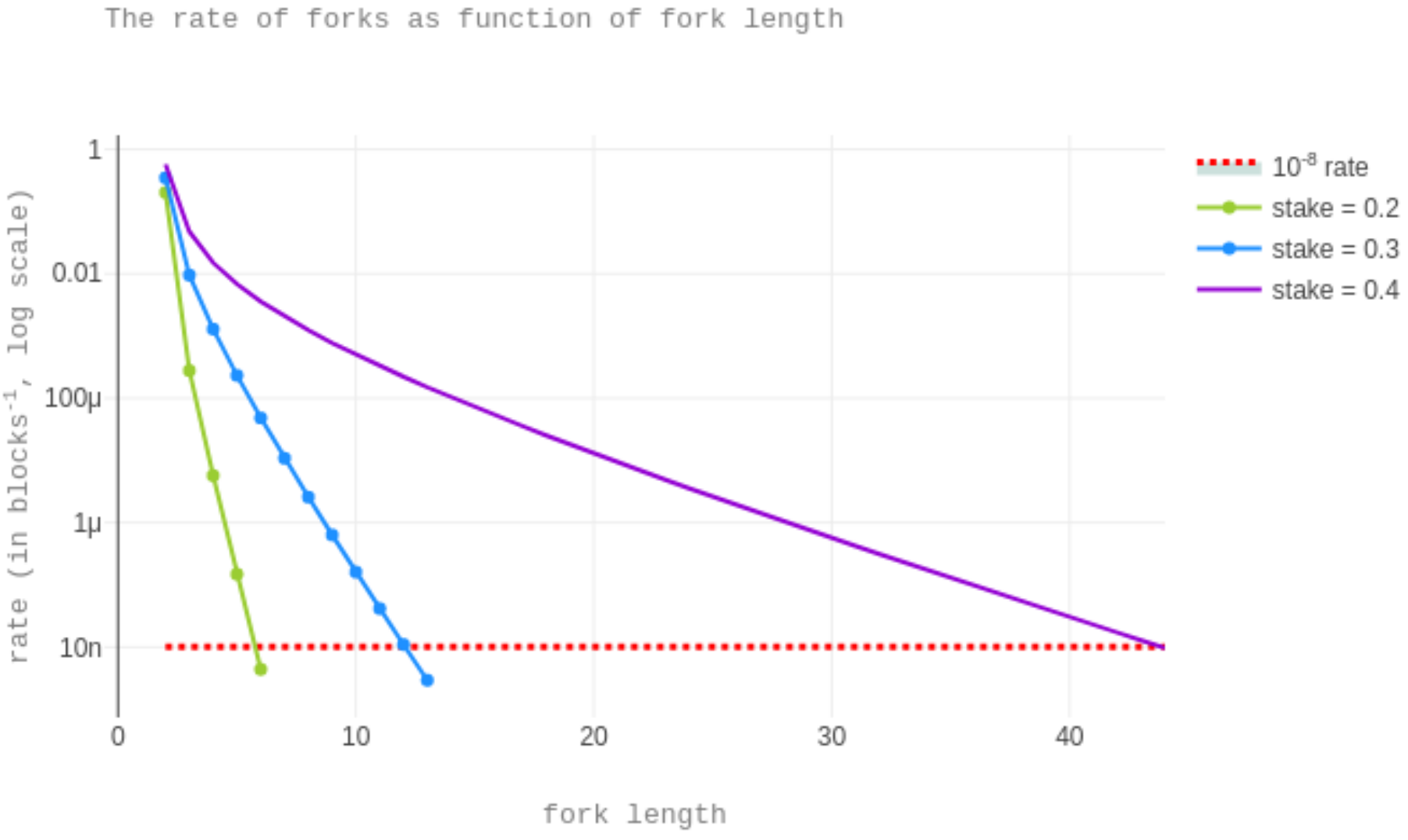
Endorsements

- An endorsement is a signature on a block.
  - included in the block at the next level
- Endorsers are selected similarly as bakers.
  - 32 endorsements slots per level
- Role
  - improve finality
  - reduce selfish baking
- Emmy$^+$ validity rule:

$$delay(blk) = 60 + 40 \cdot p + 8 \cdot \max(0, 24 - e)$$

where p is blocks' priority, e number of included endorsements

Rate of forks



Incentives

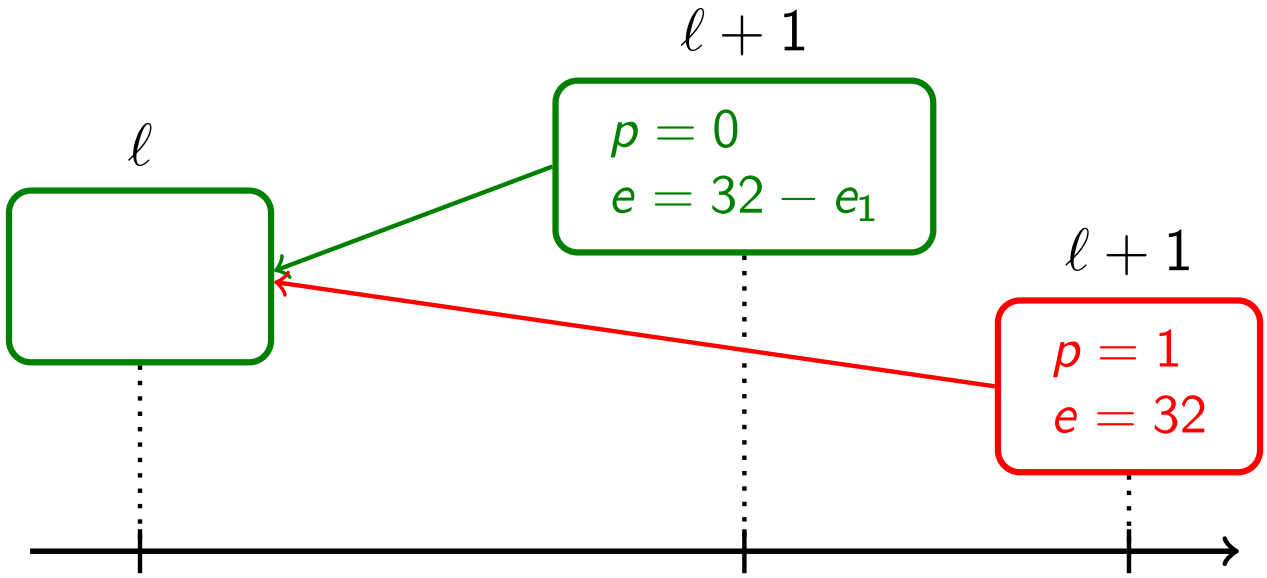
- Rational actors model is more realistic.
- Incentives and slashing are needed:
  - incentives to encourage participation
    - reward baking (16ꜩ) and endorsing (2ꜩ)
    - bakers receive the transaction fees
  - slashing to prevent double spending and nothing-at-stake problems
    - require a security deposit for baking (512ꜩ) and endorsing (64ꜩ)
    - forfeit it in case of double baking/endorsing
- Reward formulas designed to prevent two types of attacks:
  - selfish baking
  - deflationary baking

Selfish baking

- A dishonest baker D tries to "steal" blocks to gain more rewards.
- D's strategy is to withhold his endorsements.
- E.g. D has e_1 endorsements. D can steal the block if:

$$delay(1, 32) < delay(0, 32 - e_1) \iff e_1 > 12$$

  * Stealing feasible, but not necessarily profitable.



Selfish baking profitability

- It is not profitable to steal one block.
- The expected profits obtained by stealing two blocks:

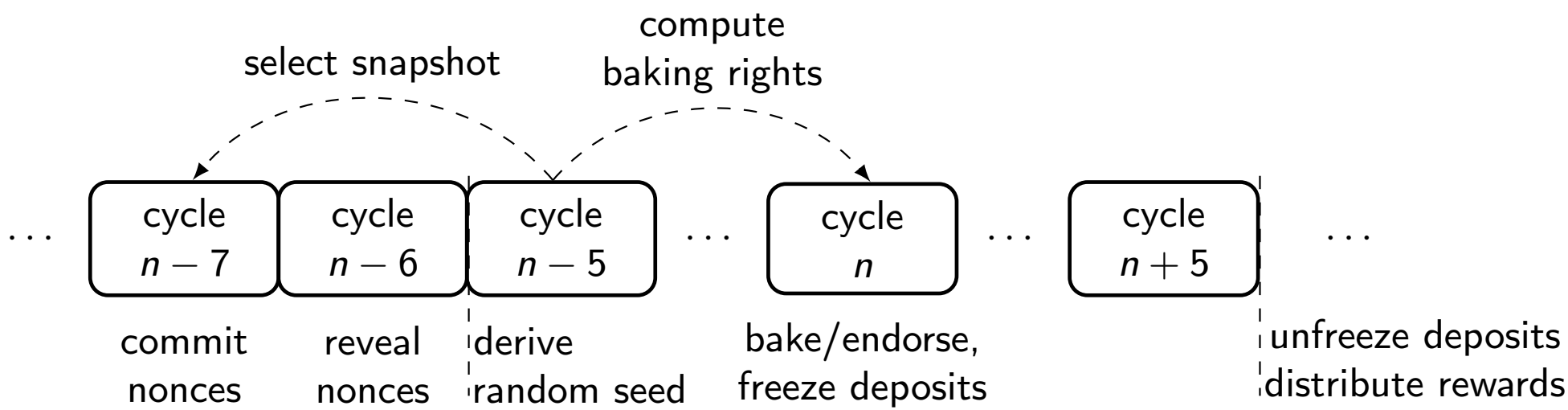
stake ratio	profits per year	percentage gain
0.10	0.26	0.000006 %
0.15	6.34	0.000101 %
0.20	43.33	0.000515 %
0.25	129.93	0.001236 %
0.30	207.86	0.001648 %
0.35	196.64	0.001336 %
0.40	115.20	0.000685 %
0.45	42.73	0.000226 %

Deflationary baking

- Issue: a baker could exclude the endorsements of others
  - if this makes the others lose more than himself
  - possible since rewards per included endorsement in Babylon are:
    - 0.1 ꜩ to baker
    - 2.0 ꜩ to endorser
- More refined notions of profitability than absolute rewards
  - Rewards adjusted to inflation
  - Rewards adjusted to stake
- Mitigation: new formulas for rewards in Carthage
  - reward per included endorsement, same for baker and endorser: 1.25 ꜩ

Consensus overview

- The protocol unfolds in cycles of 4096 blocks.
- The shell disallows any reorganization beyond 5 cycles.

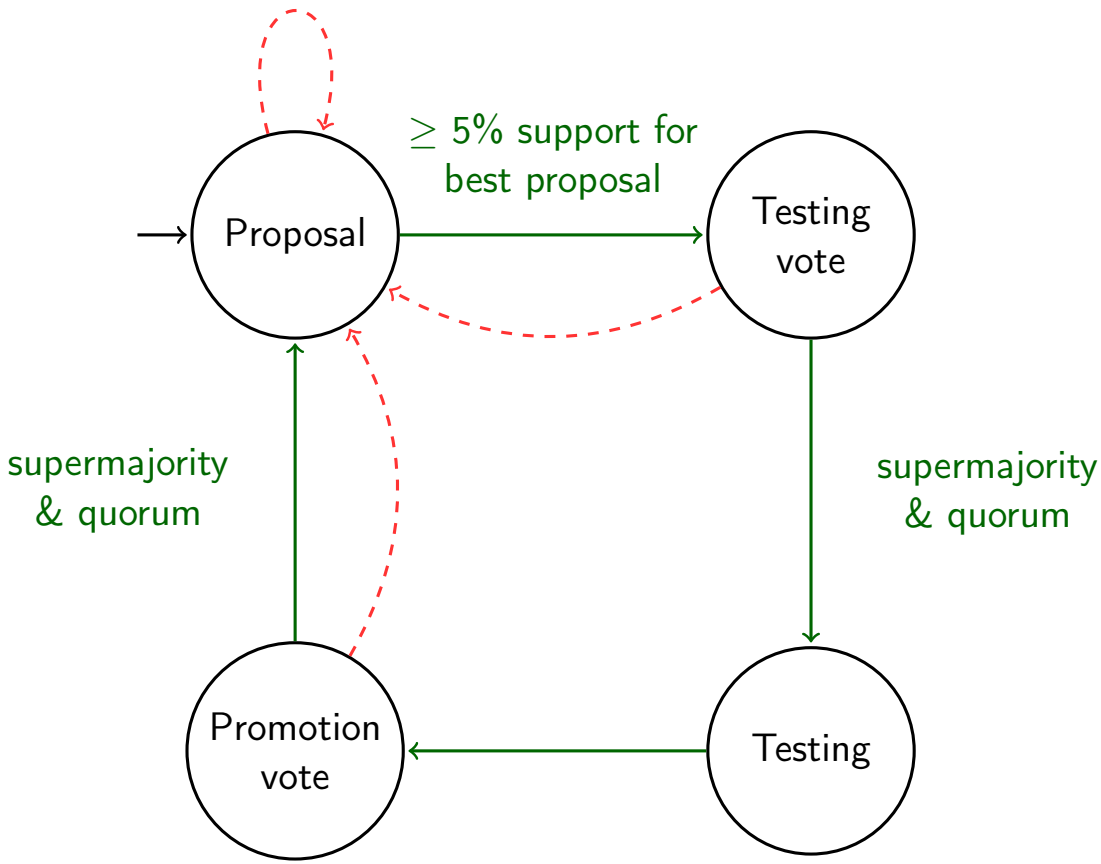


Outline

- Liquid Proof-of-Stake and Consensus
- **Protocol amendment**

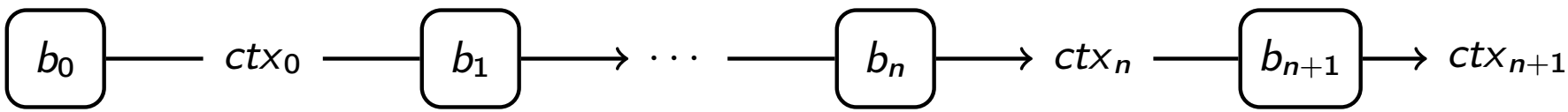
Voting procedure

- 1 vote = 1 roll; 1 voting period = 8 baking cycles $\approx$ 23 days
- supermajority: $\texttt{yays} \geq 80\%\,(\texttt{yays} + \texttt{nays})$
- quorum: $\texttt{yays} + \texttt{nays} + \texttt{passes} \geq q$



The ledger state

- blockchain = cryptographic ledger
  - blockchain = sequence of operations *and their effects*
  - Ledger state: roll owners, accounts, delegates, proposals, ballots, etc.

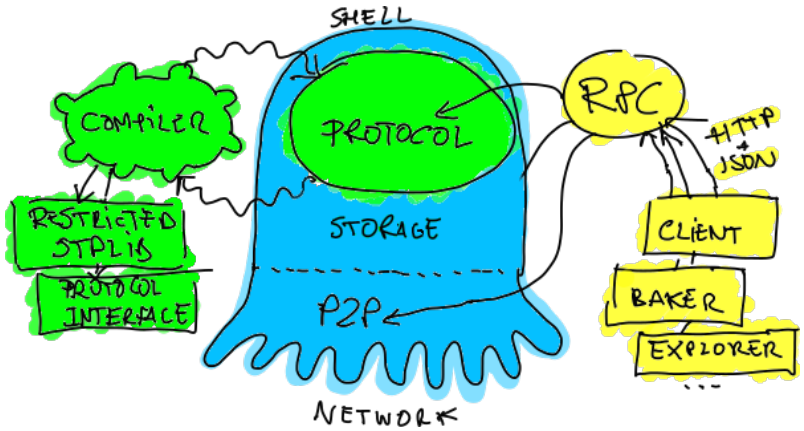


- The *context* represents the ledger state.
  - The context is a persistent, immutable key-value store.
  - Each block has an associated context, defined by the protocol.

- The protocol defines how the state is updated. It implements:

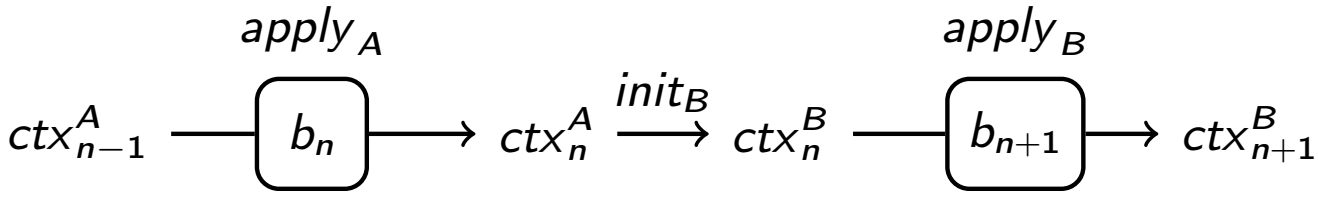
```
apply:  context -> block -> context option
```

The protocol component



- protocol = a set of OCaml modules with two constraints:
  - implements `apply`, `score`, `init`
  - it is sandboxed

Protocol activation



- Activation steps:
  - The protocol asks the shell to activate the next protocol.
  - The shell fetches the protocol.
  - The shell calls the protocol's *init* function.

Impact of protocol updates

- An update may bring
  - backward compatible improvements,
  - but also breaking changes.
- In any case:
  - Protocol developers are interested in explaining the update.
  - There is the Testing period.
  - One can influence decisions.

Take away

- Economic protocol = the rules of Tezos
- Rules can be changed through voting
  - Explicit governance so that Tezos continuously improves.
- Liquid Proof-of-Stake
  - Token owners can delegate their tokens.
  - Delegates' role is to produce blocks and to participate in governance.

Useful links

- Generic
  - Tezos developer docs: `tezos.gitlab.io`
  - Code: `gitlab.com/tezos/tezos`
  - Nomadic's blog: `blog.nomadic-labs.com`
  - Tezos stack exchange: `tezos.stackexchange.com`
  - Forums: `forum.tezosagora.org` and `www.reddit.com/r/tezos`
- Proof-of-stake and consensus algorithm
  - `tezos.gitlab.io/whitedoc/proof_of_stake.html`
  - `blog.nomadic-labs.com/analysis-of-emmy.html`
  - `blog.nomadic-labs.com/a-new-reward-formula-for-carthage.html`
- Protocol amendment and development
  - `blog.nomadic-labs.com/amendments-at-work-in-tezos.html`
  - `blog.nomadic-labs.com/how-to-write-a-tezos-protocol.html`
  - `blog.nomadic-labs.com/how-to-write-a-tezos-protocol-part-2.html`

Thank you!

Questions?