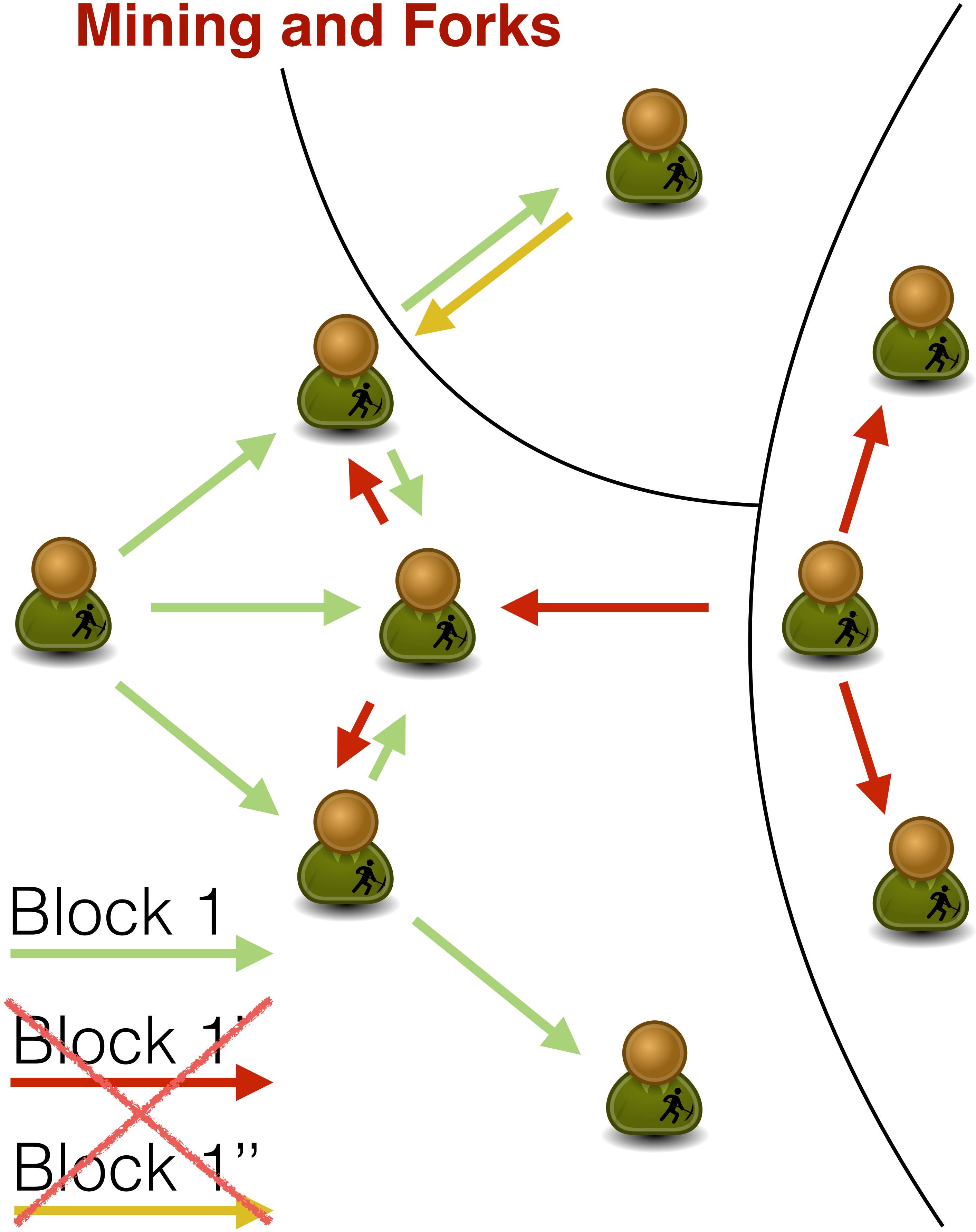
Forks 🍴

© Katerina Kamprani - The Uncomfortable

Mining and Forks



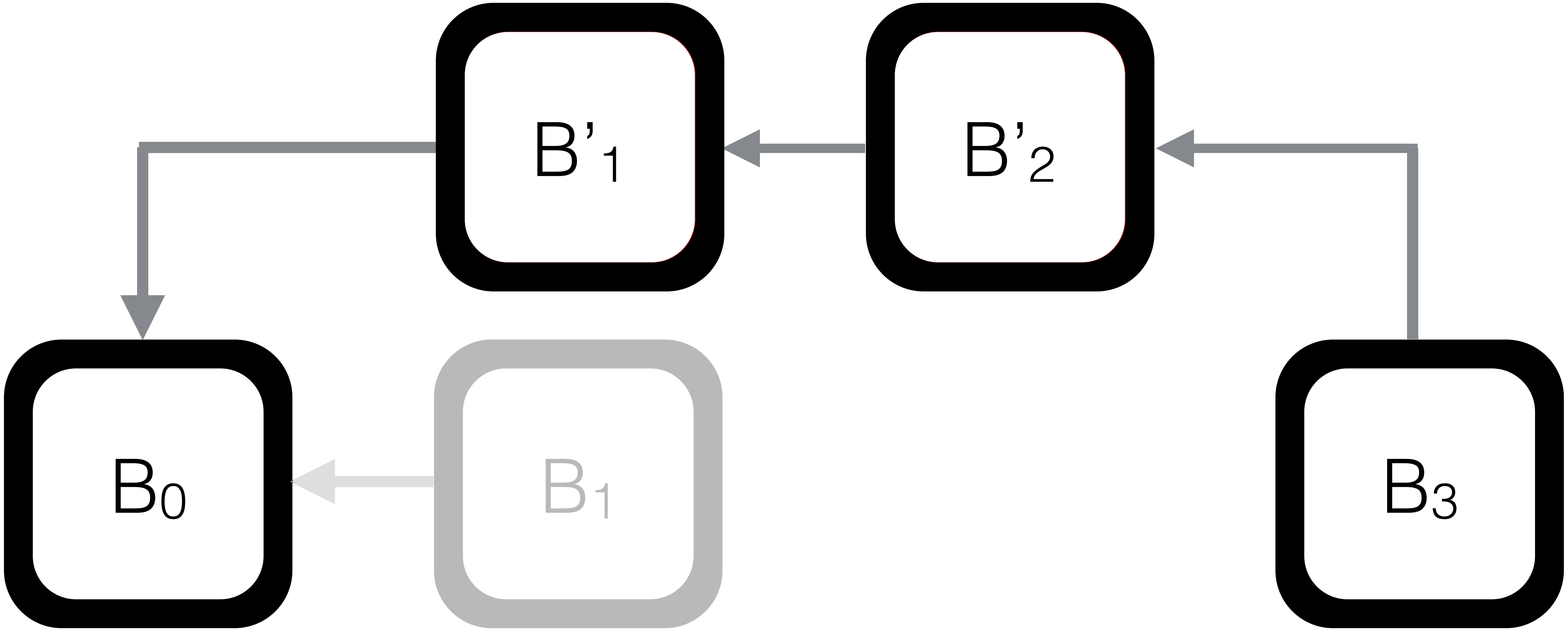
- Network partition
- **Stale blocks = lost efforts**

Selfish Mining

Denial of Service

Double Spending

Blockchain Forks



Types of Fork

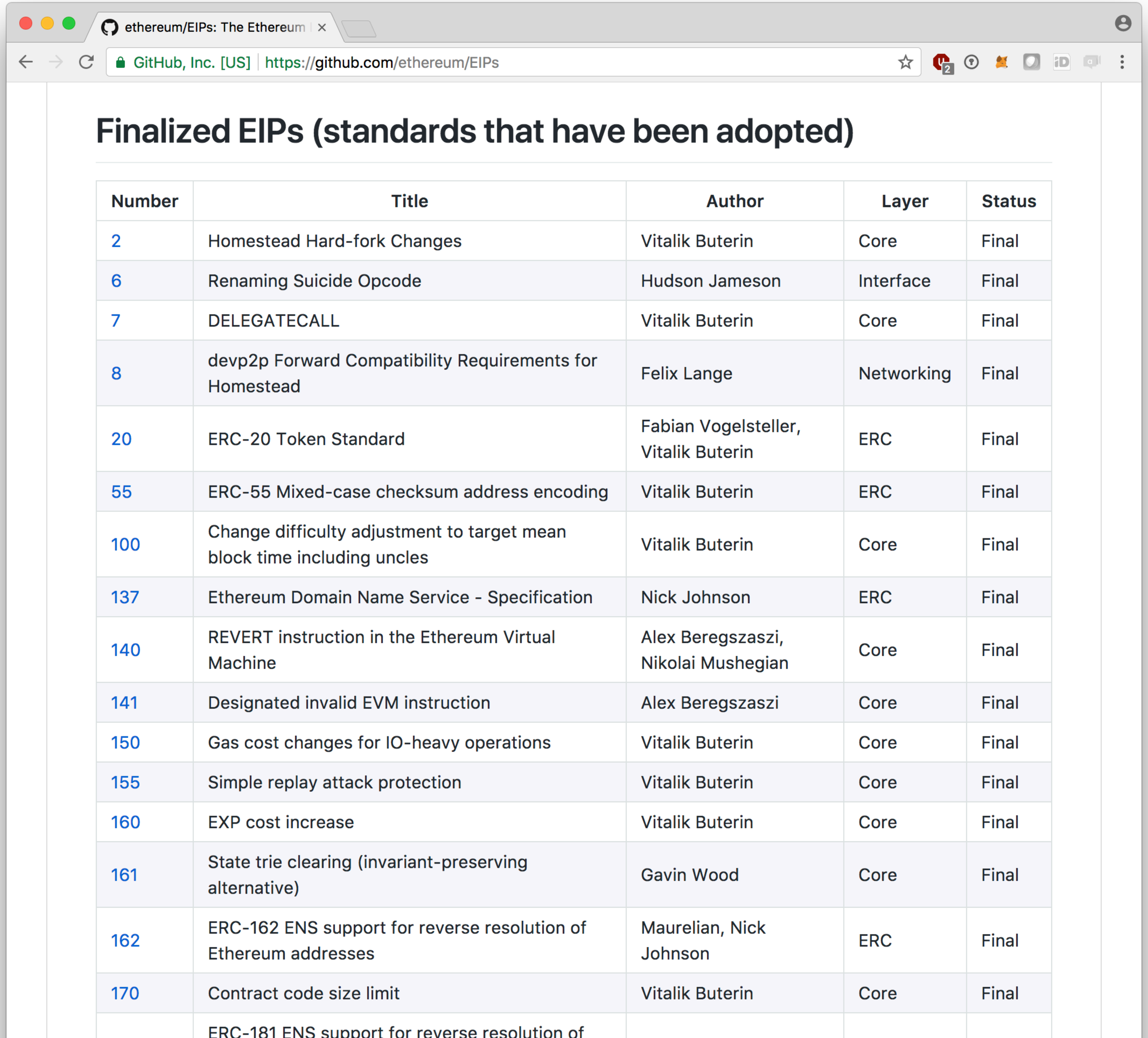
- P —> P'
  Protocol is updated from P to P'

- V —> V'
  Validity set is changed from old P to P'

- N
  The difference between V and V'

- Bitcoin Improvement Proposals (BIP)
  https://github.com/bitcoin/bips
- Ethereum Improvement Proposals (EIP)
  https://github.com/ethereum/EIPs

Ethereum Improvement Proposals



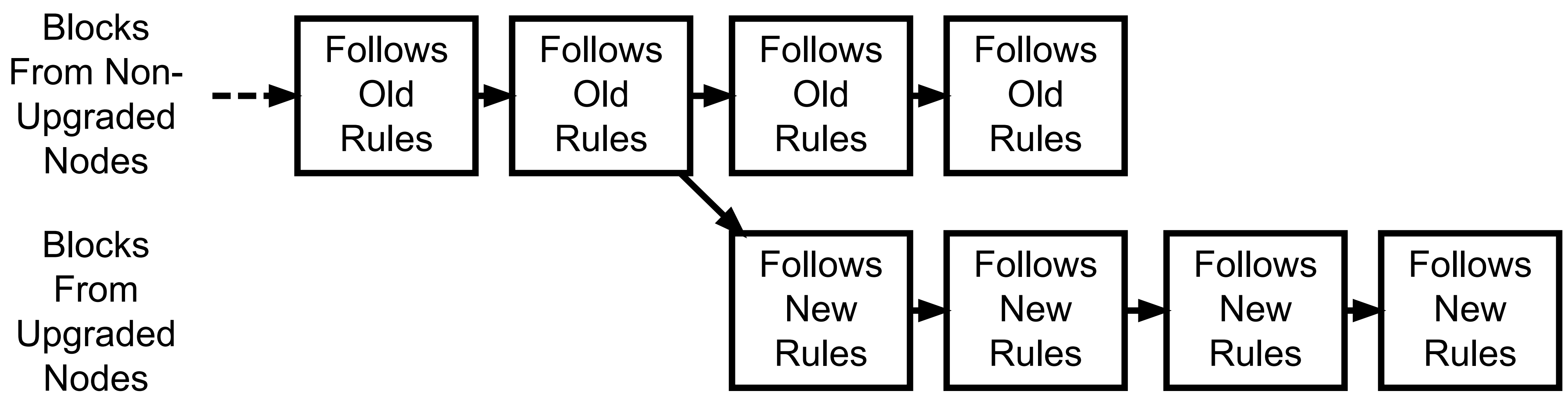
Finalized EIPs (standards that have been adopted)

Number	Title	Author	Layer	Status
2	Homestead Hard-fork Changes	Vitalik Buterin	Core	Final
6	Renaming Suicide Opcode	Hudson Jameson	Interface	Final
7	DELEGATECALL	Vitalik Buterin	Core	Final
8	devp2p Forward Compatibility Requirements for Homestead	Felix Lange	Networking	Final
20	ERC-20 Token Standard	Fabian Vogelsteller, Vitalik Buterin	ERC	Final
55	ERC-55 Mixed-case checksum address encoding	Vitalik Buterin	ERC	Final
100	Change difficulty adjustment to target mean block time including uncles	Vitalik Buterin	Core	Final
137	Ethereum Domain Name Service - Specification	Nick Johnson	ERC	Final
140	REVERT instruction in the Ethereum Virtual Machine	Alex Beregszaszi, Nikolai Mushegian	Core	Final
141	Designated invalid EVM instruction	Alex Beregszaszi	Core	Final
150	Gas cost changes for IO-heavy operations	Vitalik Buterin	Core	Final
155	Simple replay attack protection	Vitalik Buterin	Core	Final
160	EXP cost increase	Vitalik Buterin	Core	Final
161	State trie clearing (invariant-preserving alternative)	Gavin Wood	Core	Final
162	ERC-162 ENS support for reverse resolution of Ethereum addresses	Maurelian, Nick Johnson	ERC	Final
170	Contract code size limit	Vitalik Buterin	Core	Final
	ERC-181 ENS support for reverse resolution of			

Hard Fork

- Makes previously invalid blocks/transactions valid (and vice-versa)

- All nodes need to upgrade to the latest version



A Hard Fork: Non-Upgraded Nodes Reject The New Rules, Diverging The Chain

https://bitcoin.org/en/developer-guide#consensus-rule-changes

Hard Fork Example

- Ethereum forked because a smart contract vulnerability was exploited.
- The fork “reversed” the hack.
- The blockchain was no longer append-only 😥

Soft Fork

- Only previously valid blocks/transactions are made invalid —> reducing functionality

- Backwards compatible
  - Old nodes accept new blocks as valid

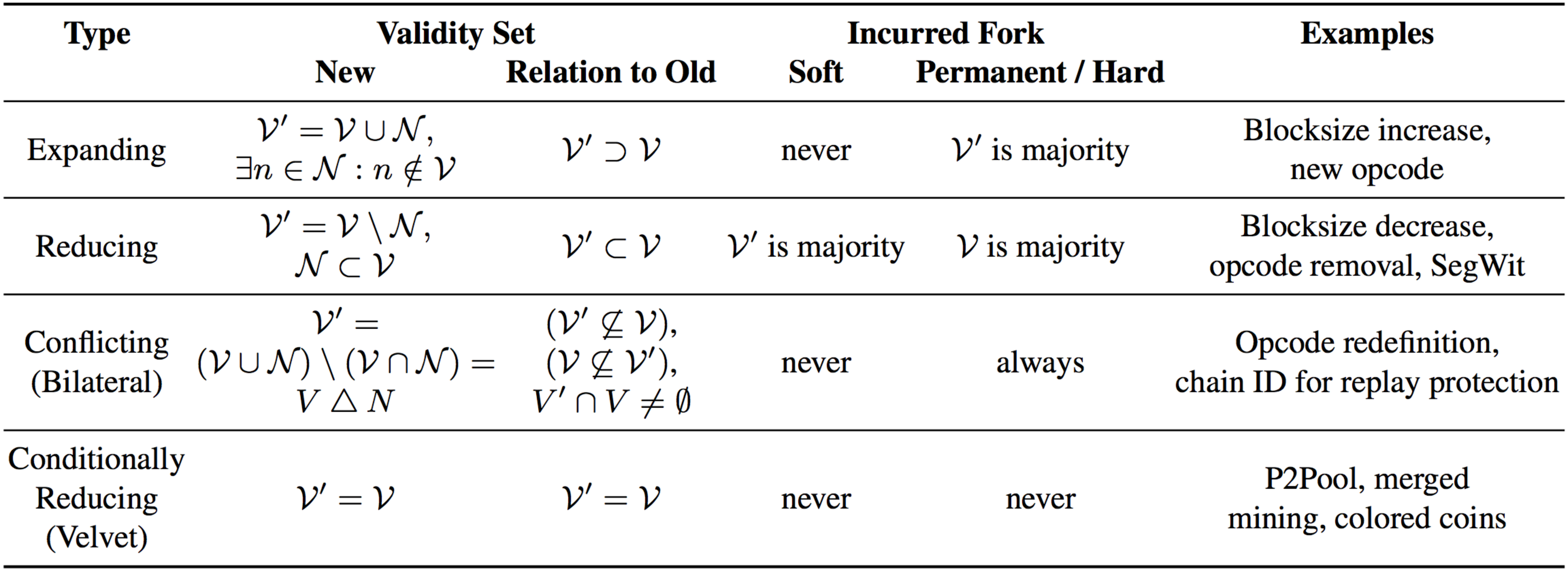
- Miner-activated Soft Fork (MASF)
  - Majority of miners upgrade to enforce

- User-activated Soft Fork (UASF)
  - Full nodes coordinate to enforce rules without miners.

Soft Fork Examples

- Make transactions >1 kb invalid
- Pay-to-script Hash (P2SH)
  - Send transactions to a Script hash (3xxxx) instead of a public key hash (1xxxx)
  - Recipient must provide a Script matching the Script hash and input data that makes the script evaluate to *true*.

Types of Fork



Type	Validity Set		Incurred Fork		Examples
	New	Relation to Old	Soft	Permanent / Hard	
Expanding	$\mathcal{V}' = \mathcal{V} \cup \mathcal{N}$, $\exists n \in \mathcal{N} : n \notin \mathcal{V}$	$\mathcal{V}' \supset \mathcal{V}$	never	$\mathcal{V}'$ is majority	Blocksize increase, new opcode
Reducing	$\mathcal{V}' = \mathcal{V} \setminus \mathcal{N}$, $\mathcal{N} \subset \mathcal{V}$	$\mathcal{V}' \subset \mathcal{V}$	$\mathcal{V}'$ is majority	$\mathcal{V}$ is majority	Blocksize decrease, opcode removal, SegWit
Conflicting (Bilateral)	$\mathcal{V}' =$ $(\mathcal{V} \cup \mathcal{N}) \setminus (\mathcal{V} \cap \mathcal{N}) =$ $V \triangle N$	$(\mathcal{V}' \not\subseteq \mathcal{V})$, $(\mathcal{V} \not\subseteq \mathcal{V}')$, $V' \cap V \neq \emptyset$	never	always	Opcode redefinition, chain ID for replay protection
Conditionally Reducing (Velvet)	$\mathcal{V}' = \mathcal{V}$	$\mathcal{V}' = \mathcal{V}$	never	never	P2Pool, merged mining, colored coins

Segregated Witness

<Sig> <PubKey> OP_DUP OP_HASH160 <PubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

- Signatures are malleable (can change while remaining valid)

- Separating transaction signature (65% of the data) from transaction data.

- Advantages
  - Increases the number of transactions that can be stored in a block.
  - Removes transaction malleability
- Disadvantages
  - Signatures required to validate a block
  - Complex

Voting through Blocks

- BIP written
- BIP discussed
- BIP implemented
- BIP voted by miners
  - Coinbase transaction contains data field
  - E.g. vote over the last 100 blocks
  - If majority (>55%) then miners implement