See previous work: Censorship rejection through "suspicion scores"

See also as background: https://vitalik.ca/general/2018/08/07/99_fault_tolerant.html

We can use the 99% fault tolerant consensus algorithm described above by having it constantly run “in the background”, to create a consensus on what data was broadcasted at what times, and therefore what data the main chain is maliciously failing to include.

Using old finalized blockchain state as a source of validators and entropy, we agree on a sequence of validators, V[1]

, V[2]

…, where each validator is assigned to a slot (eg. of 1 minute length). Let [G + D * x, G + D * (x+1)]

be the time interval associated with slot x

. At time G + D * (k * 100)

, any validator in V[k * 100] ... V[k * 100 + 99]

can publish a message (h, sig(h))

for any hash h

of a block that they saw in the previous interval of 100 * D seconds. Over the next 100 periods, the 99% fault tolerant consensus protocol runs: any validator in that set, upon receiving a value v

with n

signatures with some hash h

that they did not yet sign off on, checks if the time is less than G + D * (k * 100 + n)

, and if it is then it publishes (v, sig(v))

with their own signature.

Clients watching this process accept a block with hash h

as having been published within a given slot if they see the block and they see a message with n

signatures before time G + D * (k * 100 + n - 0.5)

. There is a proof (see blog post) that, if the network latency assumption is correct, clients will agree on which blocks were published during what interval. Now, we can socially agree on a definition of “censorship” (eg. if a block was published during interval k

, and was not included in a chain that was published after interval k+1

, then that chain is censoring that block), and coordinate on rejecting majority chains that are censoring, and coordinate on a minority soft fork to get around the attack.