Prerequisites: Enforcing windback (validity and availability), and a proof of custody

Proposer withholding and collation availability traps

Currently, when a collator is given the ability to make a collation on a shard, they are supposed to download and verify the availability of the most recent collation bodies (called windback). Let's say that windback_periods = 25

.

Assume, for now, there are no schemes in use (like those linked above) to enforce windback. In this case, it does not seem like windback is a stable equilibrium.

Before a collator makes a new collation C, they have an incentive to windback as they risk having their collation orphaned otherwise. Essentially, future collators will choose to not build on C if they find collations in C's history are unavailable while winding back themselves, causing the original collator to lose out on their collation subsidy.

However, collators also have an incentive to skip windback if they can, as it imposes some (small) cost on them.

Let's say thatwindback_periods = 25

. Each collator can reason: "the collator in front of me will check 25 collations back, but as this includes my new collation, this only overlaps with 24 of the collations I'll check." Thus, each collator can get away with windback_periods = 24

, as they can be sure the next validators will not check as far back as the 25th. This logic can continue from 24 -> 23, 23 -> 22… all the way down to windback_periods = 0

.

Essentially, it seems to be a rationalizable strategy for validators to never windback at all. (Obviously, this ignores the fact that the software will, by default, windback.)

For one, this points to the necessity of actually enforcing windback periods, like in the linked post or some other method.

Furthermore, I wonder if there's some way to make the equilibrium-without-enforcement more stable by changing windback_periods

to some mixed strategy, where validators windback some random-ish number of collations, instead of a constant. That being said, I don't know enough GT to work through a) if this is possible, or b) what it would look like if so - so any/all thoughts appreciated