One of the core insights of Celestia is that DAS;ing (Data availabiity sampling) enough to reconstruct the data is much cheaper than gossiping the full data around. This realization should be brought into consensus itself. This post sketches an outline of using DAS + direct p2p communication to lift global consensus across 100 uniform validators, with (say) 115MB/s throughput, to ~1 GB/s throughput.

This is a hasty writeup to get the idea out there, but the over-arching idea should be a massive increase in throughput. It gets Celestia consensus into the world where increasing validator counts, increases throughput. Im sure theres some p2p complexity thats relevant in parameterization thats missed here.

It also assumes we already in a world, where its possible for validators in low latency to make a proof that they have constructed a valid codeword. (e.g. just using FRI directly). It also assumes validators can communicate directly with other validators (isntead of flood gossip). It may need GPU's on validators, but not any cost increase for light nodes or full nodes.

What we should be done imo is partition our validators, such that:

- DA roots have 2 steps to finality.
- We have (say) 10 subsets of validators. Each doing Tendermint today, to reach 100MB/s of throughput on data, and publishing a DA root, along with a FRI proof that the root is of an erasure code.
- Each subset of validators publishes their view of their DA root via vote extensions.
- After a DA root is published by one validator subset, the other 9 subsets DAS sample it
- Upon receiving samples for another DA root, a validator communicates this to the entire Celestia network (via vote extensions)
- Once we know that under standard 33% BFT assumptions, the entire Celestia validator set can reconstruct any "finalized DA root", we consider that root final. In other words, there is no 1/3rd byzantine subset, who can convince you the network has the data, but actually cannot reconstruct it.

So what should happen is in a pipeline:

- At the beginning of block N, each 10% subset publishes a DA root, and gets it included as a candidate using vote extensions
- During block N voting time / block N+1 proposal time, every validator DAS samples the other shares.
- In block N+1's Vote's, they indicate which DA roots they have full data for. Upon a sufficient threshold of data arriving, they are made final.

How much can we improve throughput? Lets start parameterizing. Lets say each 10% subset requires 90% of its nodes to agree on the data (from a single proposer). Lets also assume the erasure rate is low enough for now, to give every peer a unique element of the erasure code. (This is equivalent to assuming CPU is not our bottleneck, as erasure encoding is fully parallelizable via GPU for our validators) Each subset's DA root is over SUBSET_BLOCK_SIZE

bytes of true data.

I'm going to parameterize loosely for sake of posting this. You can tighten the reasoning in several ways, and get better throughput improvement bounds. (though I'm also rounding off some terms, which may matter) I'll also assume equal stake validators for making the point.

Easiest way to do this 2/3rds dissemination, is to assume that your honest 2/3rds does not intersect with the publishing committee. So we need N bytes sent to 2/3rd's of val's. (Note every validator is receiving data, the subset is net sending 1.5 N bytes) This means each validator's data overhead increases by (SUBSET_BLOCK_SIZE * (3/2) / NUM_VALIDATORs) + DA_ROOT_AUTH_DATA

per subset, except for the subset they are part of the committee for. The DA_ROOT_AUTH_DATA

per validator can be made to be quite small, e.g. just two MT paths for a contiguous range (in the world where every val is getting independent shares). When we get to higher "share re-use" thresholds, we can get overheads here. May need to use KZG (which necissitates a GPU on just validators) to get this proof time back to negligible, when we assume erasure encoding time overheads.

So this is effectively a total bandwidth overhead of

(NUM_SUBSETs - 1) * (SUBSET_BLOCK_SIZE * (3/2) / NUM_VALIDATORs)

At 100 equal weight validators, 10 subsets, and 100MB/s of throughput per subset, we need 13.5 additional megabytes of DA sampling data, for doing a total of 10 subsets * subset block size = 1 GB/s

of DA throughput.

So this is really saying that "with 113.5 MB/s of global internet communication throughput, one more latency step, and some structured network gossip (direct peering), here is a way to achieve 1 GB/s of DA throughput

. If we already assume more structured network gossip, you can get way better sounding results here, by making each subset be more geo-local, and actually sequence more data.

There is a load bearing assumption on what is any given subset's ability to gossip data to the rest of the network. This is why I'd imagine a SUBSET_SIZE of 1 validator wouldn't work well.