There exists a directed acyclic graph (DAG) of consensus objects, each of which may have a signature and may carry data (by “carry” data we mean “has a header, that header contains a Merkle root, and if you want you can download the data corresponding to the Merkle root”; we assume that aside from this data, “consensus objects” are very small, <500 bytes).

In a simple blockchain, the DAG is just a simple chain:

But more complex structures are theoretically possible:

The directed arrows of the DAGs above actually represent two

kinds of relationships that it is very important to separate out:

- Attestation (A is an implied claim that B is valid)
- Dependency (A must be valid for B to be valid)

Dependency is transitive (if B depends on A and C depends on B, then C depends on A), attestation is not (if B attests to A and C attests to B, then C is just attesting to the fact that someone else has attested to B; they could be wrong). The clearest example of the distinction is random-sampled committee voting: if a committee votes on the validity of B, and some of the votes are wrong (ie. vote for B when it’s invalid), then those votes could still be included in the chain for economic incentivization purposes (ie. to possibly penalize the voters); the votes attest to B, but B is not a dependency of the votes. In this post, when we talk about the DAG, we will focus on dependency

.

For a client, implementing dependency checking is easy: if B1, B2, B3 are dependencies of C, then when a client sees C, it will check whether or not it has already accepted B1, B2 and B3; if it has not, then it will put C in a queue, to be checked again once B1, B2 and B3 have all been verified (and if the queue fills up, C could get popped out of the queue to save space; in this case, the client may need to just download C again).

We define the “dependency cone” of a message as either the set of messages that it depends on, or the set of messages that will depend on it (which one will generally be clear from context); it’s called a “cone” because each message by itself generally only has a few parents and a few children in the DAG, but the further one goes back through the ancestry (or descendants) the more ancestors (or descendants) there are at each level (the cone “widens”) until eventually all blocks being produced at a particular time are in the cone (and only at that time can a global Casper FFG finality loop happen to finalize all of the data, as a Casper FFG finality loop, by finalizing a common root hash, inherently acts as a chokepoint which brings everything into the dependency cone of everything else).

Blue: an object. Green: its dependency cone (going backwards). Grey: outside of its dependency cone.

In any blockchain, the client’s job is to validate the whole DAG. But there are different ways to validate any specific object in a DAG:

1. Fully download and check the data (direct verification)
2. Verify a data availability proof of the data
3. Wait for some time and verify that no fault proofs showing there is something wrong with the data have been received
4. Verify a SNARK, STARK or other purely cryptographic ZKP attesting to properties of the data
5. Verify that a committee has signed off on the data (ie. relying on honest majority assumption)

A light client effectively always does (5); an advanced light client also does (2) and (3). A full client does (1). Sharding is the art of combining these techniques so that most clients can be light clients over all the data, performing (2), (3) and (5), and possibly (4), but in the specific case where a client needs to themselves create a message (a block or a vote in a committee), allowing them to selectively perform (1), and possibly (4), on the specific subset of messages that has not yet had a sufficiently large committee sign off on the data.

In a sharded system, we assume that the bandwidth of the entire blockchain (that is, total data size of new messages coming in per second) is far higher than one single computer can process. This places bounds on how quickly dependency cones can grow: until the point in time at which the committee becomes sufficiently large to be trustworthy, the width of the dependency cone must be small enough that everything in the cone can be downloaded and checked in real time.

The design space of sharding lies in the question of, where do we put notaries, what do we make notaries notarize, and how do we structure the DAG so that viable conditional validation policies (“weakly validate if a committee of sufficient size has already signed off, otherwise strongly validate”) becomes possible?

The 1.1 chain-inside-a-chain model

The DAG looks roughly as follows:

[

%5Bpast%5D%20-%3E%20%5BM1%7Bbg%3Ablue%7D%5D%2C%5Bpast%5D%20-
%3E%20%5BS1%5D%2C%5Bpast%5D%20-%3E%20%5BS1%20%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM2%7Bbg%3Ablue%7D%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%5BM4%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM5%7Bbg%3Ablue%7D%5D%2C%5BS1%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%5BS1%20%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS2%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-%3E%20%5BS2%20%5D%2C%5BS1%5D%20-
%3E%20%5BS2%5D%2C%5BS1%20%5D%20-%3E%20%5BS2%20%5D%2C%5BS2%5D%20-
%3E%20%5BS3%5D%2C%5BS2%20%5D%20-%3E%20%5BS3%20%5D%2C%5BS3%5D%20-
%3E%20%5BS4%5D%2C%5BS3%20%5D%20-%3E%20%5BS4%20%5D%2C%5BS4%5D%20-
%3E%20%5BS5%5D%2C%5BS4%20%5D%20-%3E%20%5BS5%20%5D%2C%5BS2%5D%20-
%3E%20%5BM5%7Bbg%3Ablue%7D%5D%2C%5BS2%20%5D%20-
%3E%20%5BM5%7Bbg%3Ablue%7D%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS3%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-%3E%20%5BS3%20%5D

319×903

](https://yuml.me/diagram/scruffy/class/%5Bpast%5D%20-
%3E%20%5BM1%7Bbg%3Ablue%7D%5D%2C%5Bpast%5D%20-%3E%20%5BS1%5D%2C%5Bpast%5D%20-
%3E%20%5BS1%20%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM2%7Bbg%3Ablue%7D%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%5BM4%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM5%7Bbg%3Ablue%7D%5D%2C%5BS1%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%5BS1%20%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS2%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-%3E%20%5BS2%20%5D%2C%5BS1%5D%20-
%3E%20%5BS2%5D%2C%5BS1%20%5D%20-%3E%20%5BS2%20%5D%2C%5BS2%5D%20-
%3E%20%5BS3%5D%2C%5BS2%20%5D%20-%3E%20%5BS3%20%5D%2C%5BS3%5D%20-
%3E%20%5BS4%5D%2C%5BS3%20%5D%20-%3E%20%5BS4%20%5D%2C%5BS4%5D%20-
%3E%20%5BS5%5D%2C%5BS4%20%5D%20-%3E%20%5BS5%20%5D%2C%5BS2%5D%20-
%3E%20%5BM5%7Bbg%3Ablue%7D%5D%2C%5BS2%20%5D%20-
%3E%20%5BM5%7Bbg%3Ablue%7D%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS3%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-%3E%20%5BS3%20%5D)

S1 is a dependency of M4, but only if S2 and S3 exist and were included into or before M4; so S2 and S3 are in some sense a “dependency of the dependency relationship”. This ensures that if the committee mechanism fails, it can only fail in one direction (accepting an invalid block), the other direction does not exist because valid blocks by construction must carry committee attestations. This is critical because it means “an attestation exists AND some other check passes” can be used as a strictly more correct validity checking mechanism than just checking attestations, so the guarantees from different mechanisms can be stacked together. Notice that in a forkful model, every block (S1, S2, S3, S4, S5…) is both a data carrier and an attestation of previous messages in the same chain, and S2 and S3 form the “committee” that judges whether or not S1 is worthy of inclusion.

In reality, the gap between block creation and inclusion of a block as a dependency into the main chain would be much greater than 2; for example, it could be 100. However, this leads to our next conclusion: we can’t have a fully subjective committee threshold; some threshold must be set in-protocol

. The reason is that there must be some specific point at which S1 becomes a dependency of the main chain (and soon after a dependency of everything), and at that point we cannot ask nodes to evaluate S1 for themselves (because there is a copy of S1 for every shard, which is far too much data), so they have to trust the committee.

There are ways to construct a protocol where the threshold isn’t a hard number; for example, instead of having a single main chain, one could have a tree of chains, where a block in a chain becomes a dependency of its parent chain after N blocks; here, the dependency cone grows more slowly: N blocks of one chain after N block intervals, then adding 2N blocks from 2 chains after the next N, then adding 4N blocks from 4 chains after the next N, etc.

However, this is just setting a diagonal in-protocol threshold instead of a vertical one; for any given computer with some given internet bandwidth, eventually, the dependency cone will grow to the point where some node deciding whether or not to make a block or attestation at some particular position will not have the bandwidth to fully evaluate the entire cone, at which point they’ll be forced to rely on the implied random-sample committee of everyone who attested up until that point.

The minimal sharding model

The minimal sharding model is much simpler:

[

%2C%5BM1%7Bbg%3Ablue%7D%5D%20-%3E%20%5BS1%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM2%7Bbg%3Ablue%7D%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS1%20%5D%2C%5BS1%5D%20-%3E%20%5BM2%7Bbg%3Ablue%7D%5D%2C%5BS1%20%5D%20-
%3E%20%5BM2%7Bbg%3Ablue%7D%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS2%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS2%20%5D%2C%5BS2%5D%20-%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BS2%20%5D%20-
%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS3%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS3%20%5D%2C%5BS3%5D%20-%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%5BS3%20%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D

236×1070

](https://yuml.me/diagram/scruffy/class/%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS1%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM2%7Bbg%3Ablue%7D%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS1%20%5D%2C%5BS1%5D%20-%3E%20%5BM2%7Bbg%3Ablue%7D%5D%2C%5BS1%20%5D%20-
%3E%20%5BM2%7Bbg%3Ablue%7D%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS2%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS2%20%5D%2C%5BS2%5D%20-%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BS2%20%5D%20-
%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS3%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS3%20%5D%2C%5BS3%5D%20-%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%5BS3%20%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D)

A full-sized committee of some specific preset size (eg. 150) stands between every S[i] and the main chain block that includes it; a client need only check the committee verifications (to save further space, a client could check one committee that verifies the signatures based by each shard committee).

The 2.0 beacon chain spec

The DAG in the model roughly looks like this:

[

%5B...%5D%20-%3E%20%5BS1%5D%2C%5B...%20%5D%20-
%3E%20%5BS1%20%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS1%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS1%20%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM2%7Bbg%3Ablue%7D%5D%2C%5BS1%5D%20-%3E%20%5BS2%5D%2C%5BS1%20%5D%20-
%3E%20%5BS2%20%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS2%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS2%20%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BS2%5D%20-%3E%20%5BS3%5D%2C%5BS2%20%5D%20-
%3E%20%5BS3%20%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS3%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS3%20%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%2C%5BS1%5D%20%2B%2B-
%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BS2%20%5D%20%2B%2B-%3E%20%5BM4%7Bbg%3Ablue%7D%5D

258×568

](https://yuml.me/diagram/scruffy/class/%5B...%5D%20-%3E%20%5BS1%5D%2C%5B...%20%5D%20-
%3E%20%5BS1%20%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS1%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS1%20%5D%2C%5BM1%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM2%7Bbg%3Ablue%7D%5D%2C%5BS1%5D%20-%3E%20%5BS2%5D%2C%5BS1%20%5D%20-
%3E%20%5BS2%20%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS2%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS2%20%5D%2C%5BM2%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BS2%5D%20-%3E%20%5BS3%5D%2C%5BS2%20%5D%20-
%3E%20%5BS3%20%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS3%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-
%3E%20%5BS3%20%5D%2C%5BM3%7Bbg%3Ablue%7D%5D%20-

%3E%20%5BM4%7Bbg%3Ablue%7D%5D%2C%2C%5BS1%5D%20%2B%2B-%3E%20%5BM3%7Bbg%3Ablue%7D%5D%2C%5BS2%20%5D%20%2B%2B-%3E%20%5BM4%7Bbg%3Ablue%7D%5D)

Diamond arrows are cross-links; they commit all data in the shard chain since the last crosslink into the main chain, and must be attested to by a committee. Every shard chain block must point to a main chain block as a reference.

Signature-Linked Binary Trees

One way to greatly increase the efficiency of the data verification layer is with a series of linked committees. Consider a dependency DAG like so:

[

%5Bn0%5D%20-%3E%20%5BM%7Bbg%3Ablue%7D%5D%2C%5Bn1%5D%20-%3E%20%5BM%7Bbg%3Ablue%7D%5D%2C%5Bn00%5D%20-%3E%20%5Bn0%5D%2C%5Bn01%5D%20-%3E%20%5Bn0%5D%2C%5Bn10%5D%20-%3E%20%5Bn1%5D%2C%5Bn11%5D%20-%3E%20%5Bn1%5D%2C%5Bn000%5D%20-%3E%20%5Bn00%5D%2C%5Bn001%5D%20-%3E%20%5Bn00%5D%2C%5Bn010%5D%20-%3E%20%5Bn01%5D%2C%5Bn011%5D%20-%3E%20%5Bn01%5D%2C%5Bn100%5D%20-%3E%20%5Bn10%5D%2C%5Bn101%5D%20-%3E%20%5Bn10%5D%2C%5Bn110%5D%20-%3E%20%5Bn11%5D%2C%5Bn111%5D%20-%3E%20%5Bn11%5D

728×568

](https://yuml.me/diagram/scruffy/class/%5Bn0%5D%20-%3E%20%5BM%7Bbg%3Ablue%7D%5D%2C%5Bn1%5D%20-%3E%20%5BM%7Bbg%3Ablue%7D%5D%2C%5Bn00%5D%20-%3E%20%5Bn0%5D%2C%5Bn01%5D%20-%3E%20%5Bn0%5D%2C%5Bn10%5D%20-%3E%20%5Bn1%5D%2C%5Bn11%5D%20-%3E%20%5Bn1%5D%2C%5Bn000%5D%20-%3E%20%5Bn00%5D%2C%5Bn001%5D%20-%3E%20%5Bn00%5D%2C%5Bn010%5D%20-%3E%20%5Bn01%5D%2C%5Bn011%5D%20-%3E%20%5Bn01%5D%2C%5Bn100%5D%20-%3E%20%5Bn10%5D%2C%5Bn101%5D%20-%3E%20%5Bn10%5D%2C%5Bn110%5D%20-%3E%20%5Bn11%5D%2C%5Bn111%5D%20-%3E%20%5Bn11%5D)

Where M is a main-chain block, and at every level there is a committee. The committee does not even need to be full-sized; hypotetically, it could even be something small, like 10-40 participants. If the committee is not full-sized, clients are free to trade off between different levels of verification: if a client thinks that the committee size is 3 times less than what is acceptable, then it could personally verify the signatures going up 3 levels in the tree. This is essentially the same as the "custom windback" option in the 1.1 sharding spec, except applied to binary trees instead of unary trees (ie. chains).

I am not necessarily advocating this kind of structure immediately, as I it would be more complicated to implement and I think we can get most of the gains from one level of indirection (one main chain committee that signs off on all the other committees), but something like it is what later phases of sharding might look like.

Conclusion:

- Design of sharding protocols is design of dependency DAGs
- Width of the dependency cone at the time that some given object reaches a committee-size level of attestation is a limiting factor on speed of adding inter-shard crosslinks, which in turn is a limiting factor on reducing finality time
- We can kind of make committee sizes subjective, but it's not really

subjective (it just converts a vertical constraint into a diagonal constraint)

- It's ultimately committees attesting to committees all the way down
- However, if we stick to this paradigm where it's a DAG of objects, where every client downloads every object (hence why it's "quadratic" sharding: the number of objects can't exceed O© and the size of each object can't exceed O©), then it's fairly easy to combine together committees with data availability proofs to actually verify every object