Scalable and Probabilistic Leaderless BFT Consensus through Metastability

Traditional consensus protocols have either relied on all-to-all communication to ensure that all correct nodes reach the same decision, or a Nakamoto consensus protocol which provides a probabilistic safety guarantee. The latter is fit for open, permissionless settings; anyone can join at any time.
However, their security relies on constant participation of miner nodes. They are energy intensive, and difficult to scale with only reparameterization.

Snow is the proposed family of consensus protocols inspired by gossip algorithms.
The system correct repeatedly samples the network at random to correct nodes towards a common outcome. Analysis shows that this metastable mechanism is powerful as it can move a large network to an irreversible state quickly, where the irreversibility implies that a sufficiently portion of the network accepted a proposal, and a conflicting proposal will not be accepted with any higher than negligible probability.

Two differences between Snow- and a Nakamoto consensus is that

1. Snow protocols are green, quiescent, and efficient.
    a. Efficiency: each node requires O(1) communication overhead per round and O(log n) rounds in expectation – compared to a classical consensus that have one or more nodes requiring O(n) communication per round (phase)
2. Snow tolerates discrepancies:
    a. In contrast, classical consensus protocols require the full and accurate knowledge of n as its safety foundation.

The protocols can adopt any Sybil control mechanism, although proof-of-stake is most aligned with their quiescent operation.
Flooding attacks can be prevented with transaction fees, such as in Bitcoin, Ethereum, and the paper's proposed p2p payment system: Avalanche.

One of the algorithms of the Snow family is Snowball, used in Avalanche.
Snowball works like this (taken from the Avalanche docs):

> Everyone has an initial preference for pizza or barbecue. Until someone has *decided*, they query *k* people (the sample size) and ask them what they prefer. If α or more people give the same response, that response is adopted as the new preference. α is called the *quorum size*. If the new preference is the same as the old preference, the `consecutiveSuccesses` counter is incremented. If the new preference is different then the old preference, the `consecutiveSuccesses` counter is set to `1`. If no response gets a quorum (an α majority of the same response) then the `consecutiveSuccesses` counter is set to `0`.
>
> Everyone repeats this until they get a quorum for the same response β times in a row. If one person decides pizza, then every other person following the protocol will eventually also decide on pizza.
>
> Random changes in preference, caused by random sampling, cause a network preference for one choice, which begets more network preference for that choice until it becomes irreversible and then the nodes can decide.

Algorand

Algorand is a cryptocurrency designed to confirm transactions on the order of one minute. It uses a Byzantine agreement protocol called BA⋆ that scales to many users. This allows Algorand to reach consensus on a new block with low latency and without the possibility of forks.

Algorand faces three challenges:

1. Sybil attacks -> prevents it by adding weight to each user. BA⋆ guarantees consensus based on a weighted fraction of 2/3 honest users. Weight is determined based on the money in their account. As long as over 2/3 of the money is owned by honest users, Algorand can avoid forks and double-spending.
2. Scale to millions of users -> achieves scalability by assigning a committee. They run each step of the protocol. All other users observe. The committee is chosen randomly based on weight.
3. Resilient to DDoS attacks -> BA⋆ is a Byzantine consensus protocol that does not rely on a fixed number of servers. This prevents fixed attacks on known servers.

(PPoS)
BA⋆ algorithm for achieving consensus:

1. Block proposal: accounts are selected to propose new blocks. Each node goes through each account it manages. For each online and participating account, run Algorands verifiable random function (VRF) to confirm if or if not the account is selected to propose the block. The VRF outputs are then shared with other nodes and validated.
2. Soft Vote: each node runs the VRF for each account it manages to see if it is participating in the soft vote. Accounts are chosen based on their weight. The accounts will filter the proposals down to one by voting to confirm the block. The one with the lowest proof remains.
3. Certify Vote: a committee is selected to check the block proposal that was voted on in the soft vote phase. The VRF is executed again and votes to certify the block. If it is, it is being propagated and accumulated by each node. Once a super-majority of the votes are accumulated, the block is written to the chain.

This whole process takes less than 5 seconds.
See the video for more information and visualization:
https://www.youtube.com/watch?v=gACVKaNqxPs