Figure 1. Snow is a family of binary BFT protocols based on a non-BFT protocol *Slush*, which incrementally builds up to *Snowflake* and *Snowball* BFT protocols in the Snow family.

Figure 2. When a client creates a transaction, it names one or more parent transactions. "ancestry" represents all transactions reachable via parent edges (ancestor set), and "progeny" represents all child transactions and their offspring.

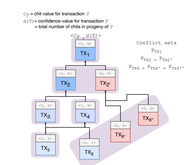

Figure 3. Transaction(s) form vertex and edges in DAG, and each transaction belongs to a conflict set in which only one can be accepted. The purple area indicates each conflict set. The chit value of a node for a transaction is 1, if and only if the node query received positive responses of `sq[quorum]` from its sampled peers. The confidence value for a transaction is the total number of chits in the transaction's progeny. The protocol accepts the one with higher confidence values.

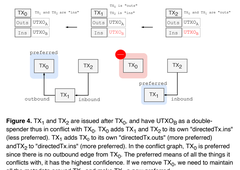

Figure 4. TX_1 and TX_2 are issued after TX_0, and have $UTXO_0$ as a double-spender thus in conflict with TX_0. TX_0 adds TX_1 and TX_2 to its own "directedTx.ins" (less preferred). TX_1 adds TX_0 to its own "directedTx.outs" (more preferred) andTX_2 to "directedTx.ins" (more preferred). In the conflict graph, TX_0 is preferred since there is no outbound edge from TX_0. The preferred means of all the things it conflicts with, it has the highest confidence. If we remove TX_0, we need to maintain all the metadata around TX_0 and make TX_1 a new preferred.

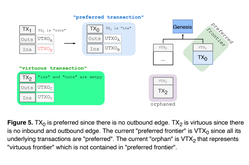

Figure 5. TX_0 is preferred since there is no outbound edge. TX_0 is virtuous since there is no inbound and outbound edge. The current "preferred frontier" is VTX_0 since all its underlying transactions are "preferred". The current "orphan" is VTX_2 that represents "virtuous frontier" which is not contained in "preferred frontier".

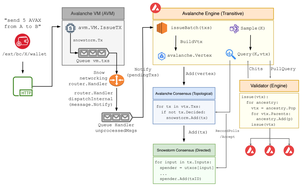

Figure 6. When the wallet send request is sent to AVM, it creates a transaction object which is then added to the queue. The Snow networking router passes those transactions to the Snow engine that creates a vertex. The Avalanche and snowstorm consensus add it to the conflict graph. Then the engine samples K validators to send the queries to. Once receiving the query, the validator responds with Chits to communicate its current preference. On receiving Chits from the validator, the querying node now collects the votes via RecordPoll and makes acceptance decisions.

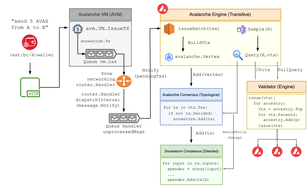

Figure 1. When the wallet send request is sent to AVM, it creates a transaction object which is then added to the queue. The Snow networking router passes those transactions to the Snow engine that creates a vertex. The Avalanche and snowstorm consensus add it to the conflict graph. Then the engine samples K validators to send the queries to. Once receiving the query, the validator responds with Chits to communicate its current preference. On receiving Chits from the validator, the querying node now collects the votes via RecordPoll and makes acceptance decisions.

Figure 4. Bitcoin block is replicated over peer-to-peer network. As soon as a new Bitcoin block is mined, the node broadcasts the block to all of its peers, and so on.

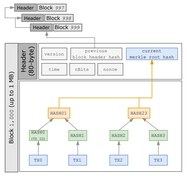

Figure 5. A Bitcoin block is mined when the node finds the "nonce" that outputs the block header hash equal to or below the "target" threshold ("nBits"). The merkle root represents the hash of all transactions in the block.

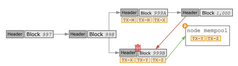

Figure 6. Two Bitcoin blocks may arrive at the same time. The node keeps both and starts extending on the first one. When the node hears or mines a new block 1,000 that was built on top of 999A, it chooses the chain with 999A and purges the other block 999B and returns the transactions in 999B but not in 999A to its mempool.