Notation for bets

Let n be the number of validators. Each validator is assigned an index from the range [1;n]

A bet is a triple $b = \langle v, (b_1, b_2, \ldots, b_n), (c_1, c_2, \ldots, c_n) \rangle$

where $v \in [1;n]$ is the validator that produced the bet b, $b_i \in [0;\infty)$ represents the highest index of a bet from the validator i that the validator v is aware of (or admits to be aware of). Finally, c_i is a set (possible empty) of fixed-size references (for example, collision-resistant hashes of) to other bets of validator $i \in [1;n]$.
In a well-formed bet:

1) If $b_v > 1$, then c_v is a non-empty set of references to the bets of the form: $b' = \langle v, (b'_1, b'_2, \ldots, b'_n), \ldots \rangle$, where $b'_v = b_v - 1$
2) If $b_v = 1$, then c_v is an empty set
3) If for $i \neq v$, $b_i > 0$, then c_i is a non-empty set of references to the bets of the form: $b' = \langle i, (b'_1, b'_2, \ldots, b'_n), \ldots \rangle$, where $b'_i = b_i$
4) If for $i \neq v$, $b_i = 0$, then c_i is an empty set

Bets are constructed in such a way that only validator v can produce a bet $b = \langle v, \ldots, \ldots \rangle$.

If in a bet $b = \langle v, (b_1, b_2, \ldots, b_n), (c_1, c_2, \ldots, c_n) \rangle$, $b_i = 0$, it means that the validator v is not yet aware (or does not admit to be aware) of any bets produced by the validator i.
A special case of a bet is when $b_v = 1$, i.e. this is the first bet, produced by the validator v. A bit of information (values 0 or 1), called the estimate, is attached to such bets. The estimates can differ from validator to validators. In fact, the purpose of the agreement protocol is to get as many validator as possible, as quickly as possible, to admit that they are aware of enough estimates to compute the “common estimate”. The “common estimate” can be decided by a weighted majority, for example.

Exchanging bets in form of views

Validators send bets to each other in the form of views. A view is a set of vets with the following two properties:

1) If a view contains a bet $b = \langle v, \ldots, (c_1, c_2, \ldots, c_n) \rangle$ and for some $i \in [1;n]$, c_i is not an empty set, then the view must also contain all bets referenced by the elements of the set c_i
2) For every bet $b = \langle v, \ldots, (c_1, c_2, \ldots, c_n) \rangle$ in a view, except one, called the tip, the view also contains another (parent) bet, $b' = \langle v,\ldots,(c'_1, c'_2, \ldots, c'_n) \rangle$, such that for some $i \in [1;n]$, the set c_i contains a reference to the bet b.

In other words, a view is a tree of bets, with a root called tip.

Irregularities (faults) in views

A view may contain three types of irregularities:

1) Equivocation. Two bets b = <i, (b_1, b_2, …, b_n), ...> and b' = <i, (b'_1, b'_2, …, b'_n), ...>, such that $b_i = b'_i = 1$, but the estimates attached to these bets, are different.
2) Multiplicity. Two bets b = <i, (b_1, b_2, …, b_n), ...> and b' = <i, (b'_1, b'_2, …, b'_n), ...>, such that $b_i = b'_i > 1$, but there exist $j \in [1;n]$, such that $b_j \neq b'_j$
3) Non-monotonicity. Two bets b = <i, (b_1, b_2, …, b_n), ...> and b' = <i, (b'_1, b'_2, …, b'_n), ...>, such that exist $j,k \in [1;n]$, where $b_j > b'_j$, but $b_k < b'_k$

If any of these irregularities occur in a view, it is possible to "convict" the validator i for not following the agreement protocol.
However, it is not necessary to refuse further communications from such validator, because it still positively contribute to the agreement. A better approach would be to de-prioritize their messages.
It also simplifies the protocol greatly, if the estimates of the convicted validators are still used in the majority calculation for the "common" estimate, for this round of agreement. Otherwise the definition of majority becomes dependent (parametric) on so-called "fault context", and it makes agreement much harder to reason about (validators will different fault contexts may arrive at different majorities even if they are all aware of the same set of estimates, therefore now one needs to make sure the fault contexts also match, which is another, much harder agreement problem).
Instead, the punishment (slashing the deposits) should be applied outside of this agreement protocol.

Redundancy in views

For any set of bets, a matrix with n rows and n columns is defined, called estimate awareness matrix (EA). Element EA[i,j] = 1, if it can be concluded from this set if bets that the validator i is aware of the estimate of the validator j. The algorithm for computing EA is as follows. Start with the initial matrix filled with 0s. For each bet b = <i, (b_1, b_2, …, b_n), ...>, set EA[i, j]=1 for all $j \in [1;n]$, such that $b_j > 0$.

For any bet b = <v, …, (c_1, c_2, …, c_n)> in a view, a span of this bet is defined recursively as a set, containing:

1) Bet b itself
2) Union of all spans of all bets referenced by non-empty elements in c_1, c_2, …, c_n

It can be easily seen that a span is also a well-formed view.

For any view, a matrix with n rows and n columns is defined, called fault awareness matrix (FA). Element FA[i,j] = 1, if it can be concluded from this view that the validator i is aware of the irregularities containing in this view, for which the validator j can be "convicted". The algorithm for computing FA is as follows. Start with the initial matrix filled with 0s. For each irregularity, take two bets b=<j,...,...> and b'=<j,...,...> that define that irregularity. Find the closest common ancestor of b and b' in the view. For all bets b''=<i,...,...> on the path from

this closest common ancestor to the tip of the view (inclusive both ancestor and the tip), set FA[i,j] = 1.

A view contains redundancy, if there are two bets in it, b = <i, (b_1, b_2, …, b_n), ...> and b' = <i, (b'_1, b'_2, …, b'_n), ...>, such that $b_i = b'_i+1$, and EA(span(b)) = EA(span(b')), and FA(span(b)) = FA(span(b')).
In other words, bet b does not add any new awareness information on top of b'.

Using redundancy to restrict messages

If views that contain redundancies are treated as invalid (note that views containing irregularities are still valid, because they carry information about faults and awareness of these faults), then it can be shown that for given number of validators n, the number of possible instantiation of the agreement protocol is finite. It follows from the intuition that every valid message (view) from the same validator always has to add more information (add at least one 1 into matrix EA or matrix FA), and there is finite number of possible instances of equivocation, multiplicity and non-monotonicity.

Thoughts of the irregularities

Out of the 3 types of irregularities, equivocation is the worst one. Is there a way to reduce its effect, or eliminate it completely?

Bet amounts instead of binary estimates.

It is not very clear to me why binary bets are considered and how the Casper betting protocol would be constructed from this. What if instead of estimates included into the bets with index 1, and further bets simply being aggregators and witnesses of the previous bets, we put the actually betting "payload" in each? Meaning that a bet becomes a tuple:
b = <v, (b_1, b_2, …, b_n), (c_1, c_2, …, c_n), blockHash, amount>
Where blockHash is the candidate block the validator is betting on (lets say, each of the block creators is allowed to propose one block per height, and the rest of the validators are choosing which one wins), and amount - is the amount taken off the deposit if some other block gets finalised.
Nice thing about this is that if there is no equivocation here - the bets simply pile up like a snowball.
Matrix EA then becomes 3-dimensional, and non-boolean. The elements of the matrix are amounts, and the 3rd dimension is the block candidate dimension. Multiplicity and Non-monotonicity are still faults, and we have binary matrix FA as before.
If bets have some granularity (perhaps non-linear), then we also get finite number of possible states of the protocol.