Polkadot Finality Gadget v9000

September 25, 2018

1   Introduction

We consider the question of finality for blockchain protocols: when will a block be reverted. Many such protocols, such as the original blockchain, Bitcoin, have the property of eventual consensus - that an ever growing prefix of the chain will be agreed upon by all participants forever onward. But they generally only give probabilistic finality on a specific block - that under some assumptions about the network and participants, if we see a few blocks building on a given block, we can estimate the probability that it is final.

But what we'd prefer is to have provable finality - for example a signed statement by a set of authorities, the set of whom ca be tracked, that the block is final. This is useful to prove what happened to light clients, who do not have the full chain or are not actively listening to the network, and to communicate with other chains, possibly as part of a scalability solution, whee not anyone receives or stores all the data in the system.

Another popular consensus mechanism for blockchains is to get Byzantine agreement on each block. This gives provable finality immediately. However this is slow if we have a large set of participants in the Byzantine agreement.

The approach that we will take is similar to the approach that Ethereum plans to take with Casper the Friendly Finality Gadget(Casper FFG)[1], which combines these approaches. We will use a block production mechanism and chain selection rule that give eventual consensus and then add a finality gadget, a protocol that finalises blocks that the participants already agree on, to get provable finality.

We present a finality gadget that works in a partially synchronous network model, GRANDPA, as well as an asynchronous finality gadget, that can cope with 1/5 Byzantine guys.

1.1   Formalising the problem

We need to incorporate into the definition of Byzantine agreement that we have access to a protocol that would achieve eventual consensus if we did not affect it. Consider a typical definition of multi-values Byzantine agreement. We have a set of participants V, most of which obey the protocol, but a constant fraction may be Byzantine i.e. behave arbitrarily

Definition 1.1. *A protocol for multi-valued Byzantine agreement has a set of values S, a a set of voters V, a constant fraction of which may be Byzantine, each of whom start with an initial value $s_v \in S$ for each $v \in V$ and in the end each voter decides a final value $f_v \in S$ such that the following holds:*

Agreement: All honest voters decide the same value for f_v Termination: All honest voters eventually decide a value Validity: If all honest voters have the same initial value, then they all decide that value

We cam change this definition to assume that instead of having an initial value, all voters have access to an external protocol, an oracle for values, that achieves eventual consensus in that it returns the same value to all voters when called after some time.

Definition 1.2. *A protocol for multi-valued Byzantine finality gadget problem has a set of values S, a a set of voters V, a constant fraction of which may be Byzantine, each of whom has access to an oracle A with the property that and in the end each voter decides a final value $f_v \in S$ such that the following holds:*

Agreement: All honest voters decide the same value for f_v Termination: All honest voters eventually decide a value Validity: All honest voters decide a value that A returned to some honest voter sometime.

Note that, in the case $|S| > 2$, this definition of validity is stronger than that the obvious generalisation for Multi-valued Byzantine agreement, that all honest voters decide a value that some honest voter started with. This is because this would be impossible if the fraction of Byzantine voters is bigger than $1/|S|$ as we cannot detect Byzantine voters who act like honest voters except for lying about their initial value so if fewer than $1/|S|$ voters act like they have some initial value, the protocol cannot know if any are honest.

But for the case $|S| = 2$, the two possible definitions of validity are equivalent. This means that we can reduce the binary version of the Byzantine finality gadget problem above to binary Byzantine agreement by each voter just calling A at the start to obtain their initial value since if A does not return the same value to every honest voter all the time then it returns both values to honest voters some times. Thus there are many existing algorithms for the binary Byzantine finality gadget problem. However the interesting problem in this case is whether the celebrated impossibility result of [2] generalizes to this finality gadget problem i.e. whether this oracle which is guaranteed to achieve eventual consensus makes it possible to have an asynchronous and deterministic protocol for agreement. A reduction is not immediately obvious. It turns out that the finality gadget version is indeed impossible see ?.

Now how do we extend this to agreeing on a chain of blocks? We will need the block production mechanism to build on finalised blocks, so the best chain rule must include them. We assume a kind of conditional eventual consensus. If we keep building on our last finalised block B and don't finalise any new blocks, then eventually we have consensus on a longer chain than just B, which the finality gadget can use to finalise another block. We also want a protocol that does not terminate, but instead keeps on finalising more blocks.

Definition 1.3. *A protocol for the blockchain Byzantine finality gadget problem has a a set of voters V, a constant fraction of which may be Byzantine, each of whom has access to an oracle for the best chain given the last finalised block with the property that, as long as no new block is finalised, it achieves eventual consensus on some child of the last finalised block such that the following holds:*

Safety: All honest voters finalise the same block at each block number Liveness: All honest voters keep finalising blocks Validity: If an honest voter finalises a block B then that block was seen in the best chain observed by some honest voter containing some previously finalised ancestor of B

1.2 Our approach

To come up with a solution to the blockchain Byzantine finality gadget problem, we will typically look at various Byzantine agreement protocols and use those to find protocols for the multi-valued Byzantine finality gadget problem. Protocols for that with appropriate properties can used to find protocols for the blockchain Byzantine finality gadget problem by considering running them in parallel at every block number. If the one block protocol has the right properties then they will agree on blocks consistently so if we finalise a block then we also finalise its ancestors and we can come up with a succinct protocol.

For example, suppose we have a one block protocol that calls for a vote on blocks which requires a participant to observe a supermajority, say votes from 2/3 of voters, for some block (or else the participant observes that the vote is undecided). Now imagine running this vote in parallel for every block number and have any honest voter vote for blocks from a particular chain. Byzantine voters may vote more than once, but if we count a vote for a block as a vote for each ancestor of the block in the vote for the instance of the one block protocol with its number, then Byzantine voters must also vote for chains, though they can vote for multiple chains. If we do this, then we see that if a block has a supermajority in a vote, then so does all its ancestors in their votes. Thus the blocks with a supermajority form a chain. Furthermore, if only 1/3 of voters equivocate then from if a participant sees a subset of th votes for chains, then they must see a prefix of the chain of blocks that all the votes have supermajorities for. Intuitively, the protocol can agree on the prefix that 2/3 of voters agree on using this.

2 Preliminaries

We will want to change the set of participants who actively agree sometimes. To model this, we have a large set of participants who follow messages. For each voting step, there is a set of n voters. We will frequently need to assume that for each such step, at most $f < n/3$ voters are faulty. We need $n - f$ of voters to agree on finality. Whether or not block producers ever vote, they will need to be participants who track the state of the protocol.

Participants remember which block they see as currently being the latest finalised block and a chain they are locked to. This locked chain represents an estimate of which block could have been finalised already.

Rounds: each participant has their own idea of what the current round number is. Every prevote and precommit has an associated round number. Honest voters only vote once (for each type of vote) in each round and don't vote in earlier rounds after later ones.

Each round has two phases, each of which has an associated vote, prevote and precommit.

For block B, we write chain(B) for the chain whose head is B. The block number, $n(B)$ of a block B is the length of chain(B).

For blocks B', B, B is later than B' if it has a higher block number. We write $B > B'$ or that B is descendant of B' for B, B' appearing in the same blockchain with B' later i.e. $B \in$ chain(B') with $n(B') > n(B)$ and $B < B'$ or B is an ancestor of B' for $B' \in$ chain(B) with $n(B) > n(B')$. $B \geq B'$ and $B \leq B'$ are similar except allowing $B = B$. We write $B \sim B'$ or B and B' are on the same chain if $B < B', B = B'$ or $B > B'$ and $B \nsim B'$ or B and B' are not on the same chain if there is no such chain.

Blocks are ordered as a tree with the genesis block as root. So any two blocks have a common ancestor but two blocks not on the same chain do not have a common descendant.

A vote v for a block B by a validator V is a message signed by V containing the blockhash of B and meta information like the round numbers and the type of vote.

We call a set S of votes tolerant if the number of validators with more than one vote in S is at most f. We say that S has supermajority for a block B if the set of validators with votes for blocks $\geq B$ has size at least $(n + f + 1)/2$.

The 2/3-GHOST function $g(S)$ takes a set S of votes and returns the block B with highest block number such that S has a supermajority for B. If there is no such block, then it returns 'nil'. (if $f \neq \lfloor (n-1)/3 \rfloor$, then this is a misnomer and we may change the name accordingly.)

Note that, if S is tolerant, then we can compute $g(S)$ by starting at the genesis block and iteratively looking for a child of our current block with a supermajority, which must be unique if it exists. Thus we have:

Lemma 2.1. *Let T be a tolerant set of votes. Then*

1. *The above definition uniquely defines $g(T)$*

2. *If $S \subseteq T$ has $g(S) \neq nil$, then $g(S) \leq g(T)$.*

3. *If $S_i \subseteq T$ for $1 \leq i \leq n$ then all non-nil $g(S_i)$ are on a single chain with head $g(T)$.*

Note that we can easily update $g(S)$ to $g(S \cup \{v\})$, by checking if any child of $g(S)$ now has a supermajority.

3 tells us that even if validators see different subsets of the votes cast in a given voting round, this rule may give them different blocks but all such blocks are in the same chain under this assumption.

We say that it is possible for a set S to have a supermajority for B if $2f + 1$ validators either vote for a block $\ngeq B$ or equivocate in S. Note that if S is tolerant, it is possible for S to have a supermajority for B if and only if there is a tolerant $T \supseteq S$ that has a supermajority for B.

We say that it is impossible for any child of B to have a supermajority in S if S has votes from at least $2f + 1$ validators and it is impossible for S to have a supermajority for each child of B appearing on the chain of any vote in S. Again, provided S is tolerant, this holds if and only if for any possible child of B, there is no tolerant $T \subset S$ that has a supermajority for that child.

Note that it is possible for an intolerant S to both have a supermajority for S and for it to be impossible to have such a supermajority under these definitions, as we regard such sets as impossible anyway.

Lemma 2.2. (i) If $B' \geq B$ and it is impossible for S to have a supermajority for B, then it is impossible for S to have a supermajority for B'.

(ii) If $S \subseteq T$ and it is impossible for S to have a supermajority for B, then it is impossible for T to have a supermajority for B.

(iii) If $g(S)$ exists and $B \nsim g(S)$ then it is impossible for S to have a supermajority for B.

3 Algorithm

We let $V_{r,v}$ and $C_{r,v}$ be the sets of prevotes and precommits respectively received by v from round r at the current time.

We define $E_{r,v}$, v's estimate of what might have been finalised in round r, to be the last block in the chain with head $g(V_{r,v})$ that it is possible for $C_{r,r}$ to have a supermajority for. If either $E_{r,v} < g(V_{r,v})$ or it is impossible for $C_{r,v}$ to have a supermajority for any children of $g(V_{r,v})$,, then we say that (v sees that) round r is completable. $E_{0,v}$ is the genesis block (if we start at $r = 1$).

We have a time bound T, that we hope is enough to send messages and gossip them to everyone.

In round r an honest validator v does the following:

1. v can start round $r > 1$ when round $r - 1$ is completable and v has cast votes in all previous rounds where they are a voter. Let $t_{r,v}$ be the time we start round r.

2. At time $t_{r,v}$, if v is the primary of this round and has not finalised $E_{r-1,v}$ then they broadcast $E_{r-1,v}$. If thy have finalised it, they can broadcast $E_{r-1,v}$ anyway (but don't need to).

3. If v is a voter for the prevote of round r, v waits until either it is at least time $t_{r,v} + 2T$ or round r is completable, then broadcasts a prevote. They prevote for the head of the best chain containing $E_{r-1,v}$ unless we received a block B from the primary and $g(V_{r-1,v}) \geq B > E_{r-1,v}$, in which case they use the best chain containing B instead.

4. If v is a voter for the precommit step in round r, then they wait until $g(V_{r,v}) \geq E_{r-1,v}$ and one of the following conditions holds

   (i) it is at least time $t_{r,v} + 4T$,

   (ii) round r is completable or

   (iii) it is impossible for $V_{r,v}$ to have a supermajority for any child of $g(V_{r,v})$, and then broadcasts a precommit for $g(V_{r,v})$ ((iii) is optional, we can get away with just (i) and (ii)).

3.1 Finalisation

If, for some round r, at any point after the precommit step of round r, we have that $B = g(C_{r,v})$ is later than our last finalised block and $V_{r,v}$ has a supermajority, then we finalise B. We may also send a commit message for B that consists of B and a set of precommits for blocks $\geq B$ (ideally for B itself if possible see "Alternatives to the last blockhash" below).

To avoid spam, we only send commit messages for B if we have not receive any valid commit messages for B and its descendants and we wait some time chosen uniformly at random from $[0, 1]$ seconds or so before broadcasting.

If we receive a valid commit message for B for round r, then it contains enough precommits to finalise B itself if we haven't already done so, so we'll finalise B as long as we are past the precommit step of round r.

4   Analysis

4.1   Accountable Safety

The first thing we want to show is asynchronous safety if we have at most f Byzantine validators:

Theorem 4.1. *If the protocol finalises any two blocks B, B' that have valid commit messages sent are on the same chain, then there are at least $f + 1$ Byzantine voters who all voted in a particular vote. Furthermore, there is a synchronous procedure to find such a set.*

The challenge procedure works as follows: If B and B' are committed in the same round, then the union of their precommits must contain at least f equivocations, so we are done. Otherwise B was committed in round r and B' in round $r' > r$ say. Then we ask the at least $n - f$ validators who precommitted $\geq B'$ in round r in the commit message, why they precommitted.

We ask queries of the following form:

- Why was $E_{r''-1} \not\geq B$ when you prevoted for or precommitted to $B'' \not\geq B$ in round $r'' > r$?

Which any honest validator should be able to respond to as is shown in Lemma 4.2 below.

The response is of the following form:

- A either a set S of prevotes for round $r'' - 1$ or a set S of precommits for round $r'' - 1$ or such that it is impossible for S to have a supermajority for B.

If no validator responds, then we have $n - f$ Byzantine validators. If any do, then if $r'' > r + 1$, we can ask the same query for $n - f$ validators in round $r'' - 1$.

If any responded and $r'' = r + 1$, then we have either a set S of prevotes or precommits in round r that it is impossible for S to have a supermajority for B in round r.

If S is a set of precommits, then if we take the union of S and the set of precommits in the commit message for B, then the resulting set of precommits for round r has a supermajority for B and it is impossible for it to have a supermajority for B. This is possible if the set is not tolerant and so there must be at least $f + 1$ voters who equivocate an so are Byzantine.

If we get a set S of prevotes for round r that does not have a supermajority for B, then we need to ask a query of the form

- Which prevotes for round r have you seen?

to all the voters of precommit in the commit message for B who voted for blocks $B'' \geq B$. There must be $n - f$ such validators and a valid response to this query is a set T of prevotes for round r with a supermajority for B'' and so a supermajority for B.

If any give a valid response, by a similar argument to the above, $S \cup T$ will have $f + 1$ equivocations.

So we either discover $f + 1$ equivocations in a vote or else $n - f > f + 1$ voters fail to validly respond like a honest voter could do to a query.

Lemma 4.2. *An honest validator can answer the first type of query.*

We first need to show that for any prevote or precommit in round r cast by an honest validator v for a block B'', at the time of the vote we had $B'' \geq E_{r-1,v}$. Prevotes should be for the head of a chain containing either $E_{r-1,v}$ or $g(V_{r,v})$ Since $g(V_{r,v}) \geq E_{r-1}, v$, in either case we have $B'' \geq E_{r-1,v}$. Precommits should be for $g(V_{r,v})$ but v waits until $g(V_{r,v}) \geq E_{r-1,v}$ before precommiting so again this holds. Thus if $B'' \not\geq B$, then we had $E_{r-1,v} \not\geq B$. Next we need to show that if we had $E_{r-1,v} \not\geq B$ at the time of the vote then we can respond to the query validly. If B wasn't on the same chain with $g(V_{r-1,v})$, then by Lemma 2.2 (iii), it was impossible for $V_{r-1,v}$ to have a supermajority for B. If it was on the same chain as $g(V_{r,-1v})$, then it was on the same chain as $E_{r-1,v}$ as well. Since $E_{r-1,v} \not\geq B$, in this case we must have $B > E_{r-1,v}$. However, possibly using that round $r - 1$ is completable, it was impossible for $C_{r-1,v}$ to have a supermajority for any child of $E_{r-1,v}$ on the same chain with $g(V_{v,r})$ and in particular for the child of $E_{r-1,v}$ on chain(B). By Lemma 2.2 (i), this means $C_{r-1,v}$ did not have a supermajority for B.

Thus we have that, at the time of th vote, for one of $V_{r-1,v}$, $C_{r-1,v}$, it was impossible to have a supermajority for B. The current sets $V_{r-1,v}$ and $C_{r-1,v}$ are supersets of those at the time of the vote, and so by Lemma 2.2 (ii), it is still impossible. Thus v can respond validly.

5

This is enough to show Theorem 1. Not that if v sees a commit message for a block B in round r and has that $E_{r',v} \not\geq B$, for some completable round $r' \geq r$, then they should also be able to start a challenge procedure that successfully identifies at least $f + 1$ Byzantine voters in some round. Thus we have that:

Lemma 4.3. *If there at most f Byzantine voters in any vote, B was finalised in round r and an honest participant v sees that round $r' \geq r$ is completable, then $E_{r',v} \geq B$.*

4.2 Liveness

We show the protocol is deadlock free and also that it finalises new blocks quickly in a weakly synchronous model.

Let's define $V_{r,v,t}$ be the set $V_{r,v}$ at time t and similarly for $C_{r,v,t}$ and the block $E_{r,v,t}$.

Lemma 4.4. *Let v, v' be (possibly identical) honest participants, t, t' be times and r be a round. Then if $V_{r,v,t} \subseteq V_{r,v',r'}$ and $C_{r,v,t} \subseteq C_{r,v',r'}$, all these sets are tolerant and v sees that r is completable at time t, then $E_{r,v,t} \leq E_{r,v',t'}$ and v' sees that r is completable at time t'.*

Proof. Since v sees that r is completable at time t, $V_{r,v,t}$, $C_{r,v,t}$ each contain votes from $n - f$ voters and so the same holds for $V_{r,v',t'}$ and $C_{r,v',t'}$. By Lemma 2.1, $g(V_{r,v',t'}) \geq g(V_{r,v,t})$. Using Lemma 2.2, since it is impossible for $C_{r,v,t}$ to have a supermajority for any children of $g(V_{r,v,t}$, it is impossible for $C_{r,v',t'})$ as well and so $E_{r,v',t'} \leq g(V_{r,v,t})$. But now $E_{r,v,t}, E_{r,v',t'}$ are the last blocks on chain$(g(V_{r,v,t}))$ that it is possible for $C_{r,v,t}, C_{r,v',t'}$ respectively to have a supermajority for. Thus by Lemma 2.2 (ii), $E_{r,v',t'} \leq E_{r,v,t}$. □

4.2.1 Deadlock Freeness

Now we can show deadlock freeness for the asynchronous gossip network model, when a message that is sent or received by any honest participant is eventually received by all honest participants.

Proposition 4.5. *Suppose that we are in the asynchronous gossip network model and that at most f voters for any vote are Byzantine. Then the protocol is deadlock free.*

Proof. We need to show that if all honest participants reach some vote, then all of them eventually reach the next.

If all honest voters reach a vote, then they will vote and all honest participants see their votes. We need to deal with the two conditions that might block the algorithm even then. To reach the prevote of round r, a participant may be held up at the condition that round $r - 1$ must be completable. To reach the precommit, a voter may be held up by the condition that $g(V_{r,v}) \geq E_{r-1,v}$.

For the first case, the prevote, let S be the set of all prevotes from round $r - 1$ that any honest voter saw before they precommitted in round $r - 1$. By Lemma 2.1, when voter v' precommitted, they do it for block $g(V_{r-1,v'}) \leq g(S)$. Let T be the set of precommits in round r cast by honest voters. Then or any block $B \not\leq g(S)$, T does not contain any votes that are $\geq B$ and so it is impossible for T to have a supermajority for B. In particular, it is impossible for T to have a supermajority for any child of $g(S)$.

Now consider a voter v. By our network assumption, there is a time t by which they have seen the votes in S and T. Consider any $t' \geq t$. At this point we have $g(V_{r,v,t;}) \geq g(S)$. It is impossible for $C_{r,v,t'}$ to have a supermajority for any child of $g(S)$ and so $E_{r-1,v,t'} \leq g(S)$, whether or not this inequality is strict, we satisfy one of the two conditions for v to see that round $r - 1$ is completable at time t'. Thus if all honest voters reach the precommit vote of round $r - 1$, all honest voters reach the prevote of round r.

Now we consider the second case, reaching the precommit. Note that any honest prevoter in round r votes for a block $B_v \geq E_{r-1,v,t_v}$ where t_v is the time they vote. Now consider any honest voter for the precommit v'. By some time t', they have received all the messages received by each honest voter v at time t_v and v''s prevote. Then by Lemma 4.3, $B_v \geq E_{r-1,v,t_v} \geq E_{r-1,v',t'}$. Since $V_{r,v',t'}$ contains these B_v, $g(V_{r,v',t'}) \geq E_{r-1,v',t'}$. Thus if all honest voters prevote in round r, eventually all honest voters precommit in round r.

An easy induction completes the proof of the proposition. □

4.2.2 Weakly synchronous liveness

Now we consider the weakly synchronous gossip network model. The idea that there is some global stabilisation time(GST) such that any message received or sent by an honest participant at time t is received by all honest participants at time $\max\{t, \text{GST}\} + T$.

Let t_r be the first time any honest participant enters round r i.e. the minimum over honest participants v of $t_{r,v}$.

Lemma 4.6. *Assume the weakly synchronous gossip network model and that each vote has at most f Byzantine voters. Then if $t_r \geq GST$, we have that*

(i) $t_r \leq t_{r,v} \leq t_r + T$ for any honest participant v,

(ii) no honest voter prevotes before time $t_r + 2T$,

(iii) any honest voter v precommits at the latest at time $t_{r,v} + 4T$,

(iv) for any honest v, $t_{r+1,v} \leq t_r + 6T$.

Proof. Let v' be one of the first validators to enter round r i.e. with $t_{r,v'} = t_r$. By our network assumption, all messages received by v' before they ended are received by all honest participants before time $t_r + T$. In particular at time t_r, v' sees that all previous rounds are completable and so by Lemma 4.3, so does every other honest validator by time $t_r + T$. Also since for $r' < r$, at some time $s_{r'} \leq t_r$ $g(V_{r',v',s'_r}) \geq E_{r',v',s'_r}$, again by Lemma 4, for all honest v, $g(V_{r',v,t_r+T}) \geq E_{r',v,t_r+T}$. Looking at the conditions for voting, this means that any honest validator does not need to wait before voting in any round $r' \leq r$. Thus they cast any remaining votes and enter round r by time $t_r + T$. This shows (i).

For (ii), note that the only reason why an honest voter would not wait until time $t_{r,v} + 2T \geq t_r + 2T$ is when $n - f$ voters have already prevoted. But since some of those $n - f$ votes are honest, this is impossible before $t_r + 2T$

Now an honest voter v'' prevotes at time $t_{r,v''} + 2T \leq t_r + 3T$ and by our network assumptions all honest validators receive this vote by time $t_r + 4T$. An honest voter for the precommit v has also received all messages that v'' received before they prevoted by then. Thus the block they prevoted has $B_{v''} \geq E_{r-1,v''} \geq E_{r-1,v,t_r+4T}$, since this holds for every honest voter v'', $g(V_{r,v,t_r+4T}) \geq E_{r-1,v,t_r+4T}$. Thus they will precommit by time $t_{r,v} + 4T$ which shows (iii).

By the network assumption an honest voter v''s precommit will be received by all honest validators v by time $t_{r,v'} + 5T \leq t_r + 6T$. Since v will also have received all prevotes v say when they precommitted by this time, their vote $B_{v'}$ will have $B_{v'} = g(V_{r,v'}) \leq g(V_{r,v,t_r+6T})$. Thus C_{r,v,t_r+6T} contains precommits from $n - f$ voters v' with $B_{v'} \leq g(V_{r,v,t_r+6T})$ and thus it is impossible for C_{r,v,t_r+6T} to have a supermajority for any children of $g(V_{r,v,t_r+6T})$. Thus v sees that round r is completable at time $t_r + 6T$. Since they have already prevoted and precommitted if they were a voter, they will move to round $r + 1$ by at latest $t_t + 6T$. This is (iv). □

Lemma 4.7. *Suppose $t_r \geq GST$ and very vote has at most f Byzantine voters. Let H_r be the set of prevotes ever cast by honest voters in round r. Then*

(a) any honest voter precommits to a block $\geq g(H_r)$,

(b) every honest participant finalises $g(H_r)$ by time $t_r + 6T$.

Proof. For (a), we separate into cases based on which of the conditions (i)-(iii) that we wait for to precommit hold.

For (i), all honest voters prevote in round r by time $t_r + 3T$. So any honest voter v who precommits at or after time $t_{r,v} + 4T \geq t_r + 4T$ has received all votes in H_r and by Lemma 2.1, precommits to a block $\geq g(H_r)$.

For (ii), we argue that no honest voter commits a block $\not\geq g(H_r)$ first. The result will then follow by an easy induction once the other cases are dealt with. Suppose that no honest voter has precommitted a block $\not\geq g(H_r)$ so far and that a voter v votes early because of (ii).

Note that, since we assume that all precommits by honest voters so far were $\geq g(H_r)$, it is possible for $C_{r,v}$ to have a supermajority for $g(H_r)$. For (ii) to hold for a voter v i.e for round r to be completable, it must be the case that either it is impossible for $C_{r,v}$ to have a supermajority for $g(V_{r,v})$ or else be impossible for $C_{r,v}$ to have a supermajority for any children of $g(V_{r,v})$. By Lemma 2.2 cannot have $g(V_{r,v}) < g(H_r)$. But by Lemma 2.1, these are on the same chain and so $g(V_{r,v}) \geq g(H_r)$. Since this is the block v precommits to, we are done in case (ii)

For (iii), let v be the voter in question. Note that since $n - f$ honest voters prevoted $\geq g(H_r)$, it is possible for $V_{r,v}$ to have a supermajority for $g(H_r)$. By Lemma 2.1, $g(V_{r,v})$ is on the same chain as $g(H_r)$. For (iii), it is impossible for $V_{r,v}$ to have a supermajority for any children of $g(V_{r,v})$. If we had $g(V_{r,v}) < g(H_r)$, by Lemma 2.2, this would mean that it would be impossible for $V_{r,v}$ to have a supermajority for $g(H_r)$ as well. So it must be that $g(V_{r,v}) \geq g(H_r)$ as required.

For (b), combining (a) and Lemma 4.6 (iii), we have that any honest voter v precommits $\geq g(H_r)$ by time $t_{r,v} + 4T$. By our network assumption, all honest participants receive these precommits by time $t_r + 6T$ and so finalise $g(H_r)$ if they have not done so already. $\square$

Lemma 4.8. *Suppose that $t_r \geq GST$, the primary v of round r is honest and no vote has more than f Byzantine voters. Let $B = E_{r-1,v,t_{v,r}}$ be the block v broadcasts if it is not final. Then every honest prevoter prevotes for the best chain including B and all honest voter finalise B by time $t_r + 6T$.*

Proof. By Lemma 4.6 and our network assumptions, no honest voter prevotes before time $t_r + 2T \geq t_{r,v} + 2T$ and so at this time, they will have seen all prevotes and precommits seen by v at $t_{r,v}$ and the block B if v broadcast it then. By Lemma 4.4, any honest voter v' has $E_{r-1,v'} \leq B \leq g(V_{r-1,v})$ then.

So if the primary broadcast B, then v' prevotes for the best chain including B. If the primary did not broadcast B, then they finalise it. By Lemma 4.3, it must be that $E_{r-1,v'} \geq B$ and so $E_{r-1,v'} = B$ and so in this case v' also prevotes for th best chain including B.

Since all honest voters prevote $\geq B$, $g(H_r) \geq B$ and so by Lemma 4.7, all honest participants finalise B by time $t_r + 6T$ $\square$

Lemma 4.9. *Suppose that $t_r \geq GST + T$ and the primary of round r is honest. Let B be the latest block that is ever finalised in rounds $< r$ (even if no honest participant finalises it until after t_r). If all honest voters for the prevote in round r agree that the best chain containing B include the same child B' of B, then they all finalises some child of B before $t_r + 6T$.*

Proof. By Lemma 4.3, any honest participant sees that $E_{r-1} \geq B$ during round r. Let v be the primary of round r and $B'' = E_{r-1,v,t_{r,v}}$. If $B'' > B$, then by Lemma 4.8, all honest validators finalise B'' by time $t_r + 6T$ which means they finalised a child of B. If $B'' = B$, then by Lemma 4.7, all honest voters prevote for th best chain including B. By assumption these chains include B' and so $g(H_r) \geq B$. By Lemma 4.7, this means that B' is finalised by time $t_r + 6T$. $\square$

4.2.3   Recent Validity

Lemma 4.10. *Suppose that $t_r \geq GST$, the primary of round r is honest and all votes have at most f Byzantine voters. Let B be a block that no prevoter in round r saw as being in the best chain of an ancestor of B at the time they prevoted. Then either all honest validators finalise B before time $t_r + 6T$ or no honest validator ever has $g(V_{r,v}) \geq B$ or $E_{r,v} \geq B$.*

Proof. Let v' be the primary of round r and let $B' = E_{r-1,v',t_{r,v'}}$. If $B' \geq B$, then by Lemma 4.8, all honest validators finalise B by time $t_r + 6T$. If $B' \not\geq B$, then by Lemma 4.8, no honest validator prevotes $\geq B$ and so no honest validator ever has $g(V_{r,v}) \geq B$. $\square$

Corollary 4.11. *For $t - 6T > t' \geq GST$, suppose that an honest validator finalises B at time t but that no honest voter has seen B as in the best chain containing some ancestor of B in between times t' and t, then at least $(t - t')/6T - 1$ rounds in a row had Byzantine primaries.*

5 Practicalities

5.1 Changing the voter set on-chain in an asynchronously safe way

Suppose we have an on-chain protocol that decides we need a different voter set. Once everyone finalises the block, they know that we need to change the set. The protocol can cope with changing the voter set from some round r. The main difficulty is that the chain has no idea what the current round number is and even if we have a block that instructs us to change the voter set at round r, we might only finalise the block after round r. So instead we will not take advantage of the ability to change set from one round to the next.

A block B can contain an instruction that we should change to the voter set to some other set after some integer $m \geq 0$ blocks. If our best chain for a prevote contains such a block B, then we do not prevote for more than m blocks after B, even if our best chain is longer. Thus if the current voter set has $n - f$ honest voters, they will only finalise m blocks after such a B. We only accept votes and commit messages up top m blocks after B from the current set of validators.

When some block B' that is m blocks after B has been finalised, then the new validator set starts again at round 1 with $E_0 = B'$. Votes will need to contain additional metadata that indicates the validator set somehow.

5.2 Alternatives to the last block hash

The danger with voting for the last blockhash in the best chain is that maybe no one else will have seen and processed the next block. It would also be nice to make the most of BLS multisig/aggregation, which allows a single signature for many messages/signers than can be checked in time proportional to the number of different messages signed.

To get round the first alone, it might be better to vote for a block 3/4 along (rounding further) the unfinalised chain , rather than for th head.

But the second suggests that maybe we should be including signatures for several of the latest blocks in a chain. We could include that last 2 or 3. We could also do e.g. the the blocks with block numbers with the last 2 multiples of each power of two since th last finalised block, which gives log unfinalised chain length messages but should have many blocks in common.

When presented with a vote that includes many blocks, we should interpret them as being for th last block we've seen if any. Then we need to be able to update that vote to a later block when that is seen. This retains monotonicity of a supermajority for/ it is impossible to have a supermajority for over time.

It does not matter if some of the votes are for a block that does not exist as everyone will ignore that part of the vote. But including votes for block that are seen but are not on a chain is an equivocation and is slashable. We need to count such votes as votes for the had of every chain in the vote (as someone might interpret them as for any one of them).

Then if we need to BLS aggregate votes that are $\geq B$ for a commit message or query response, it is OK to use any vote that is $\geq B$, not necessarily the vote for th head. This should reduce the number of blockhashs sign, in the optimistic case down to 1.

5.3 Block production rule

If we adopt that rule that block producers should build on the best chain including the last finalised block, then if we don't finalise another block this will eventually include some prefix beyond the last finalised block, and therefore the protocol is live by Lemma 4.10.

But the issue is that if agreement is much slower than block production, then we might have a prevote for a short chain on the last finalised block, then the best chain does not include that block and we build a long chain that is eventually never finalised. This could be fixed by building on E_{r-1} or E_r. But if we do that, and these change very quickly, then we may never come to agreement on the best chain.

So we have two possible chain selection rules for block producers:

1. Build on the best chain including the last finalise block B.

2. Build on best chain including whichever of $\{E_r, E_{r-1}, B\}$ is latest and $\geq B$.

1 is better if finalisation is happening quickly compared to block production and 2 is best if block production is much faster. We could also consider hybrid rules like adopt 1 unless we see that the protocol is stuck or slow, then we switch to 2.

6 Why?

6.1 Why do we wait at the end of a round and sometimes before precommitting?

If the network is badly behaved, then these steps may involve waiting an arbitrarily long time. When the network is well behaved (after the GST in our model), we should not be waiting. Indeed there is little point not waiting to receive 2/3 of voters' votes as we cannot finalise anything without them. But if the gossip network is not perfect, an some messages never arrive, then we may need to implement voters asking other voters for votes from previous rounds in a similar way to the challenge procedure, to avoid deadlock.

In exchange for this, we get the property that we do not need to pay attention to votes from before the previous round in order to vote correctly in this one. Without waiting, we could be in a situation where we might have finalised a block in some round r, but the network becomes unreliable for many rounds and gets few votes on time, in which case we' need to remember the votes from round r to finalise the block later.

6.2 Why have a primary?

We only need the primary for liveness. We need some form of coordination to defeat the repeated vote splitting attack. The idea behind that attack is that if we are in a situation where almost 2/3 of voters vote for something an the rest vote for another, then the Byzantine voters can control when we see a supermajority for something. If they can carefully time this, they may be able to split the next vote. Without the primary, they could do this for prevotes, getting a supermajority for a block B late, then split precommits so we don't see that it is impossible for there to be a supermajority for B until late. If B is not the best block given the last finalised block but B' with the same block number, they could stop either from being finalised like this even if the (unknown) fraction of Byzantine players is small.

When the network is well-behaved, an honest primary can defeat this attack by deciding how much we should agree on. We could also use a common coin for the same thing, where people would prevote for either the best chain containing $E_{r-1,v}$ or $g(V_{r-1,v})$ depending on the common coin. With on-chain voting, it is possible that we could use probabilistic finality of the block production mechanism - that if we don't finalise a block and always build on the best chain containing the last finalised block then not only will the best chain eventually converge, but if a block is behind the head of the best chain, then with positive probability, it will eventually be in the best chain everyone sees.

In our setup, having a primary is the simplest option for this.

7 The asynchronous finality gadget problem

Here we give an extension of the [2] result that shows the impossibility of having an asynchronous and deterministic finality gadget protocol and give an asynchronous protocol that uses a common coin primitive.

7.1 Impossibility of a deterministic protocol

The asynchronous binary fault tolerant agreement problem is as follows:

We have number of voters which each have an initial v_i in $\{0,1\}$

We may have one or more faulty nodes, which here means going offline at some point. Nodes have asynchronous communication - so any message arrives but we have no guarantee when it will. The goal is to have all non-faulty nodes output the same v, which must be 0 if all inputs v_i are 0 and 1 if all are 1.

Fischer, Lynch and Paterson[2] showed that this is impossible if there is one faulty node.

The binary fault-tolerant finality gadget problem is similar, except now there is an oracle A that any node can call at any time with the following properties:

either A always outputs x in $\{0,1\}$ to all nodes at all times or else there is an x in $\{0,1\}$ and for each node i, there is a T_i such that when i calls A before T_i. it gives x but if it calls A after T_i, it returns not x .

and we want that if A never switches, then all non-faulty nodes output x. If A does switch then all non-faulty nodes should output the same thing, but it can be 0 or 1.

Then this is also impossible, even for one faulty node, which just goes offline. Note that this generalises Byzantine agreement, since if we could each node i could call A once at the start and use the output as v_i. (For the multi-valued case, we will define the problem so that this reduction does not hold.)

Proof sketch. We follow the notation of [2] and assume for a contradiction that we use a correct protocol. Let r be a run of the protocol where A gives 0 all the time. Then by correctness r decides 0. Now we consider what can happen when A switches to 1 after each configuration in r. If it switches to 1 at the start, then the protocol decides 1. If we switch to 1 when all node have already decided 0, then we decide 0.

We claim that some configuration in the run r, where there are two runs from it where A is always 1 that decide 0 and 1. We call such states 1-bivalent. Too see this, assume for a contradiction that r contains no such configurations. Then there is are successive configurations C,C' such that if A return 1 in the future from C then we always decide 0 but from C', we always decide 1. Let events be (p,m,x) where node (processor/validator) p receives message m (which my be null) and executes some code where any calls to A return x in $\{0,1\}$, then sends some messages. Then there is some event $(p,m,0)$ that when applied to C gives C'. Now suppose that p goes offline at C, then if A always returns 1 afterwards, then we still decide 1. Thus there is a run r' that starts at C where p tales no steps, A always returns 1 and all other nodes still output 1. But since p takes no steps in r', we can apply r' after $(p,m,0)$ and so we have that C' has a run where A always returns 1 but decides 1, which is a contradiction.

Now let C be a 1-bivalent configuration. We can follow the FLP proof to show that there is a run from C for which A always returns 1, all messages are delivered but all configurations are 1-bivalent and so the protocol never decides. This completes the proof by contradiction that there is no correct protocol. $\qquad\square$

7.2 1/5 BFT finality gadget using a common coin

1. Everyone prevotes for the best chain including the block they were locked to last round.

2. Wait to receive prevotes from 4/5 of validators, V_r.

3. Precommit $g_{3/5}(V_r)$

4. Wait to receive precommits from 4/5 of validators, C_r.

5. Call the common coin, s_r

6. If $s_r = 1$, finalise $g_{3/5}(C_r)$

7. lock to $g_{(3-2s_r)/5}(C_r)$ for next round.

The common coin does not return a coin until 4/5 call it. Then assuming 4/5 are honest, it returns an s_r sampled uniformly from $\{0,1\}$, identical for all who called it, and before 4/5 called it, no-one has any information about the result.

Here $g_t(S)$ is the t-GHOST function where g from subsection 2 is $g_{2/3}$. It might not be unique in general but it is here with under 1/5 Byzantine.

The idea behind the proof of asynchronous liveness is that for a particular block B', some value of the common coin, either all the validators who received 4/5 of precommits before the common coin was decided lock to B' or none do. If we had a fixed threshold for locking, an adversarial choice of the number of precommits for B' or its descendants could lead to some validators locking to it and some not (and indeed there would be runs that do this indefinitely as this is how the impossibility result works for this type of algorithm.)

Lemma 7.1. *If we there are enough precommits to finalise a block B in round r, then all honest validators who prevote in future rounds will be locked to B or its descendants when they do.*

We want to show that this is asynchronously live:

Proposition 7.2. *Suppose that block B is finalised before round r. With probability at least $1/2$ over the common coin in round r, if all validators agree that the best chain including the last finalised block B includes a decedent B', at the prevote step of rounds $r+1$ and $r+1$, then a descendant of B is finalised the next time $s_r = 1$ after round $r + 1$ or earlier.*

Proof. By the previous lemma, all honest validators prevote in round r for B or its descendants and so all honest validators precommit to B or its descendants.

Let V_r be the set of prevotes of all validators. By a lemma from the other thing, all honest validators precommit $g_{3/5}(V_r)$ or its ancestors. Thus $g_{3/5}(V_r)$ is B or a descendant of B.

For the case $g_{3/5}(V_r) = B$, all honest validators precommit B and so any honest validator sees that $B = g_{1/5}(C_r) = g_{3/5}(C_r)$. Thus all honest validators lock B and so are free to prevote for B' or its descendants in round $r + 1$. Thus we finalise B' in round $r + 1$.

Otherwise, let B'' be the child of B in the chain of $g_{3/5}(V_r)$. We seek to show that we finalise either B' or B'' in round $r + 2$ or earlier.

Let S be the set of honest validators who precommit in round r before the common coin is flipped. Let S' be the set of honest validators who call the common coin before it is decided. Note that $S' \subset S$. Since 4/5 of validators call the coin before it decided, S' and S each contain at least 3/5 of the validators.

Let h be the number of validators in S that precommit B'' or its descendants. Note that the other $h - |S|$ validators just precommit B.

Now consider a particular validator $v \in S'$ and the set C_r of precommits they received in step 4. the number of validators with precommits in C_r is at least $4n/5$. All the honest validators with precommits in C_r are in S. Thus we have that the number of votes for B'' or its descendants in C_r, m_v has $h - n/5 \le m_v < h + n/5$. Since any descendant of B that is not B'' or its descendants receives less than $n/5$ precommits for it or its descendants, we have that either $g_{1/5}(C_r) = B$ or $g_{1/5}(C_r) \ge B'$ and similarly for $g_{3/5}(C_r)$. Now note that if $h > 2n/5$, for $v \in S'$, $m_v \ge n/5$ and so $g_{1/5}(C_r) \ge B'$. On the other hand if $h \le 2n/5$, $m_v < 3n/5$ and so $g_{3/5}(C_r) = B$.

If $h > 2n/5$ and $s_r = 1$, then every honest validator locks a block $\ge B'$. Thus is round $r + 1$, thy all prevote $\ge B'$. Since every honest validator waits until receiving prevotes from 4/5 of voters, these prevotes contain vote $\ge B'$ from $3n/5$ voters and so they prevote for $g_{3/5}(V_{r+1}) \ge B$. $\square$

References

[1] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017. URL `https://arxiv.org/abs/1710.09437`.

[2] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985. URL `https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf`.