“Casper the friendly ghost”
Correct-by-construction consensus safety

By: Vlad Zamfir

February 17th, 2017

EDCON

Outline

Set up:

What is correct-by-construction protocol design?

Overview of consensus protocol basics

Binary Casper data structures + definitions

Estimate safety

Outline

Consensus safety proof

The correct-by-construction Casper consensus decision rule using an estimate safety oracle

Proof of consensus safety of the Casper consensus decision rule using an estimate safety oracle

Outline

Completing the specification

The ideal adversary as an implementation of the safety oracle

Lower bounds on safety

An ideal adversary for the lower bound on safety

Outline

Roadmap for the correct-by-construction Casper

Q + A

Conclusion

So, what is correct-by-constrution protocol design?

“The Normal Way”

1) Formally specify the protocol

2) Define properties that protocol must satisfy

3) Prove that the protocol has given properties

“The Correct-by-Construction Way”

1) Formally *but partially* specify the protocol

2) Define properties that protocol must satisfy

3) Derive more of the protocol in a way that is proven to satisfy them

“The Normal Way”

1) Formally and completely specify the protocol

2) Define properties that protocol must satisfy

3) Prove that the protocol has given properties

The goal is that giving a proofs of the protocol’s correctness is almost trivial.

Because it will have been derived specifically to satisfy these proofs.

We’re going to see a very short consensus safety proof in this talk!

But first… a bit about consensus!

Basics of Consensus

Consensus safety:

Any two protocol-following nodes, if they decide on a value, decide on the same value.

Consensus liveness:

All protocol-following nodes eventually decide on a value.

Basics of Consensus

Byzantine fault tolerant consensus:

A *bft* consensus protocol has safety and liveness even in the context of some number **t** of nodes behaving arbitrarily.

Asynchronous consensus

We can prove safety (and liveness (?)) without making assumptions about network latency and clock synchronization.

Now lets talk about Casper...

...for the binary consensus !

Basics of Binary Casper

Validators:

We have a set of “validators” V.

Weights:

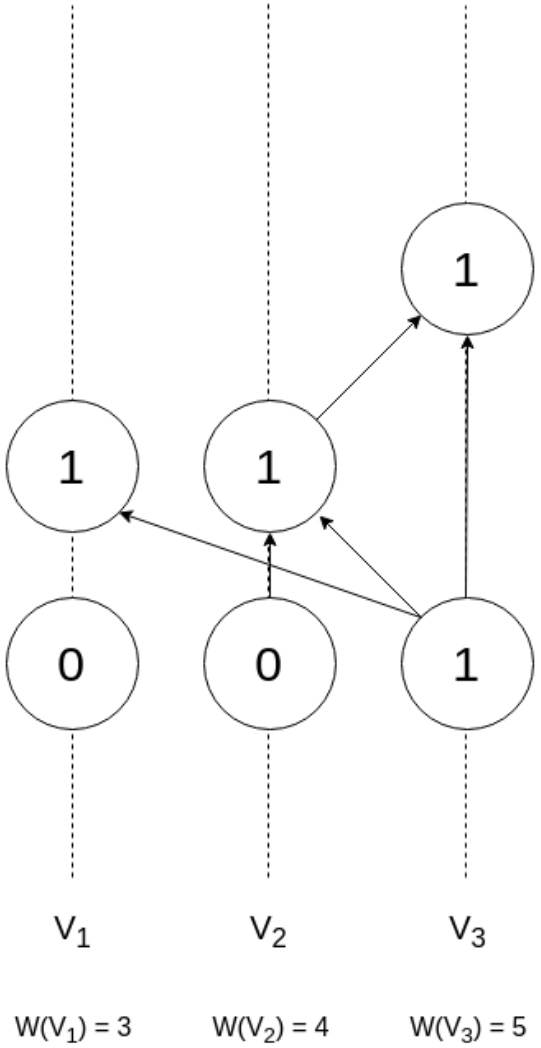
Each validator **v** has a weight W(**v**)

Bets:

Are a triple:

- estimate (0 or 1)
- sender (in V)
- justification (a set of bets)

Basics of Binary Casper



Basics of Binary Casper

Invalid bets:

Any bet that does not have an estimate that is also the “max weight” estimate in its justification (if defined)….

In a given justification:

Assuming that the most recent bets with estimate 0 come from validators with total weight W_0 (and that W_1 is analogously defined), then:

If W_0 > W_1,

then a bet with this justification is invalid if it has estimate 1.

Basics of Binary Casper

Equivocation:

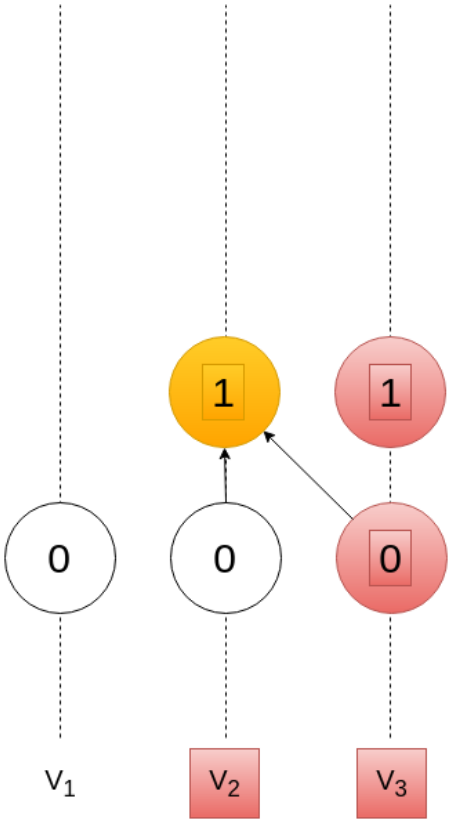
Two distinct bets such that:

- neither are a dependency of the other
- they are from the same sender

A view in non-bft Casper:

Any set of bets without any equivocations

Or invalid bets.



And now… the safety definition!

Estimate safety in the binary Casper

Estimate safety:

An estimate is safe in a given view if all possible futures of that view have the same max-weight estimate.

Estimate safety oracle:

Is able to efficiently compute whether an estimate is safe.

Casper consensus decision rule

Estimate safety decision rule:

A protocol-following node decides on a value if and only if it is safe according to its estimate safety oracle given its view.

We will now prove the asynchronous consensus safety of this decision rule!

Proof of consensus safety

Suppose that a node has decided e1, given a view u1

And that another has decided e2, given a view u2

Decisions only happen if the nodes have estimate safety, so we know that:

Safe(e1,u1) and Safe(e2,u2)

Proof of consensus safety

We know also that u1 U u2 is a possible future of u1

(u1 U u2 is assumed not to contain equivocations!)

It follows that Safe(e1,u1) => Estimate(u1 U u2) = e1

Proof of consensus safety

We know also that u1 U u2 is a possible future of u2

(u1 U u2 is still assumed not to contain equivocations!)

It follows that Safe(e2,u2) => Estimate(u1 U u2) = e2

So: e1 = Estimate(u1 U u2) = e2
…which is what was to be shown!

The need for an ideal adversary

To complete the specification of the protocol, we need to implement a estimate safety oracle.

An “ideal adversary” against an estimate in a view u:

- Returns a possible future of u violating the estimate
- Or raises an exception/does not return a future of u.

Returning a value if and only if the estimate is not safe

The need for an ideal adversary

The ideal adversary does a search of possible futures of a given view for views that violate an estimate...

...but does not need to search all possible futures.

Implementing the ideal adversary is, at is core, a dynamic programming problem.

A strategy for the ideal adversary

The ideal adversary (with victim estimate 0) searches possible futures by repeatedly attempting to create latest bets with estimate 1 and showing them to validators with latest estimate 0 until she succeeds or stops making progress.

A strategy for the ideal adversary

The “side effect” of showing a validator a bet is showing it all of the bets in its justification + dependency.

The ideal adversary would be easier to compute if there were no side effects.

Lower bounds on safety

Lower bound estimate safety oracles:

If LB oracle says we safe then we really are safe

(If LB oracle says we are not safe then maybe we are not safe)

We can still enjoy the safety proof if we replace estimate safety oracles with lower bound oracles!

Lower bounds on safety

Now the crazy part:

The ideal adversary on Casper without side effects provides a lower bound on safety of Casper.

Margin is unfortunately too narrow to include the proof

Lower bounds on safety

But roughly..

The side effects of showing a bet in Casper are free options to an ideal adversary in Casper without side effects.

This lack of side effects gives the ideal adversary strictly more ability to find futures that violate an estimate.

Thus Casper without side effects is less safe than Casper.

Lower bounds on safety

So we can calculate a lower bound on safety with the following ideal adversary:

```
while(progress_made):
    progress_made = false
    for v in validators_voting_against_attacker:
      try:
        make_latest_bet_with_given_estimate(v, target_estimate)
        progress_made = true
      except:
        continue
```

And now we’ve done all the work!

The correct-by-contruction decision rule

Now we are able to give the derived protocol:

Nodes decide on a value given a view only when they find that an estimate with that value is safe against the ideal adversary for the given view in Casper without side effects.

Recall the safety proof, concluding in e1 = Estimate(u1 U u2) = e2

We must show only that the ideal adversary implements a lower bound on safety, to benefit!

Notice that the consensus safety proof was given before the specification was complete!

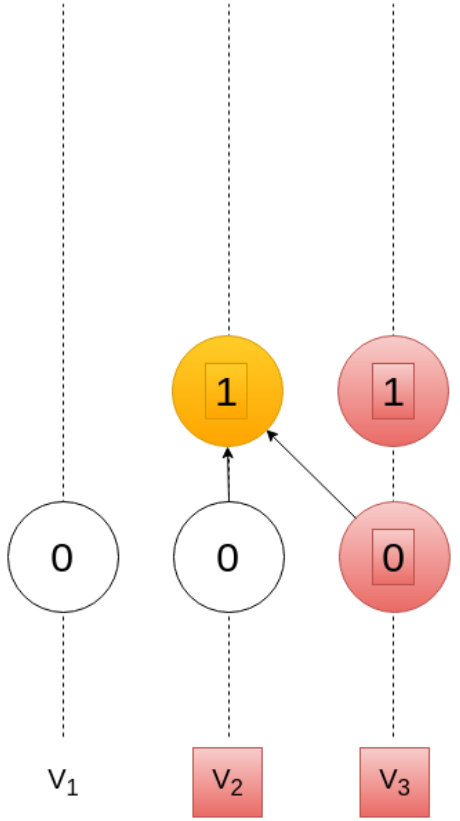
Maybe a lot of the work is “hidden” in choosing the model and definitions. Who knows!

A roadmap for CbC Casper

From Binary Consensus…

To light-client friendly BFT EVM replication…

…with mechanism design!



A roadmap for CbC Casper

Asynchronous Binary Consensus

- Uses a fixed set of validators with fixed weights.
- Decides on a single bit.
- Safety is a binary decision (safe or not safe).
- Is safe in asynchronous networks.
- Is live in synchronous network.

A roadmap for CbC Casper

Asynchronous BFT Binary Consensus

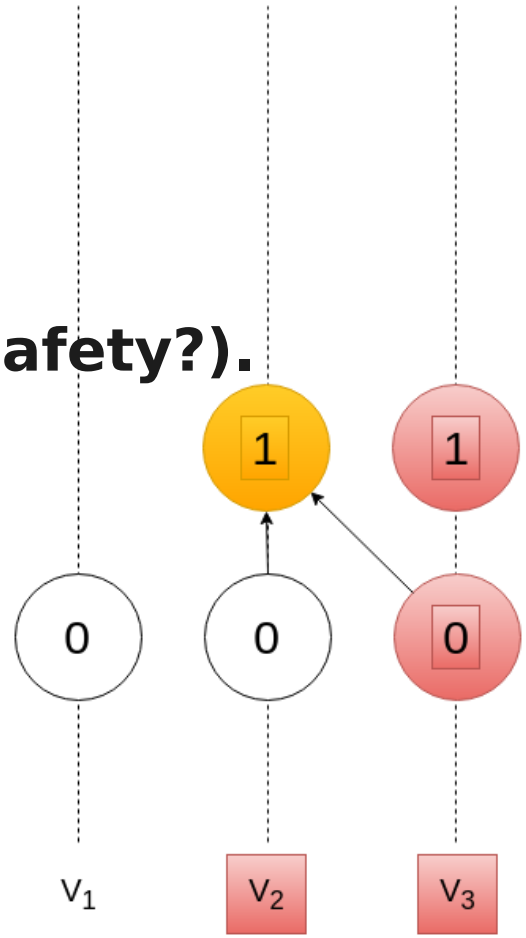
Uses a fixed set of validators.

Decides on a single bit.

Safety is a threshold decision (how *much* safety?).

Is safe in asynchronous networks.

Is live in synchronous network.



A roadmap for CbC Casper

Asynchronous BFT EVM Replication

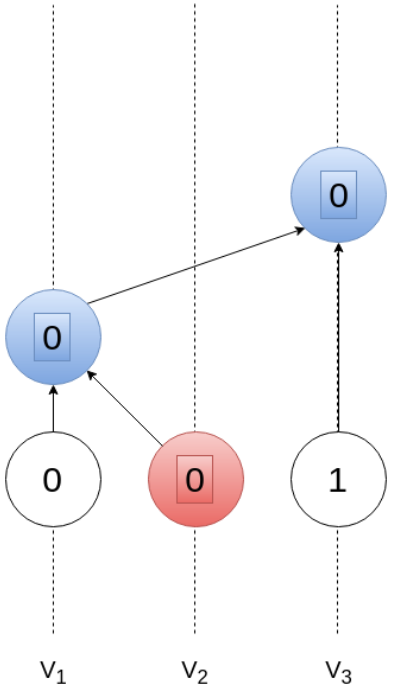
Uses a fixed set of validators.

Decides on the state of the EVM.

Safety is a threshold decision (how *much* safety?).

Is safe in asynchronous networks.

Is live in synchronous network.



A roadmap for CbC Casper

Asynchronous BFT EVM Replication

...with validator rotation

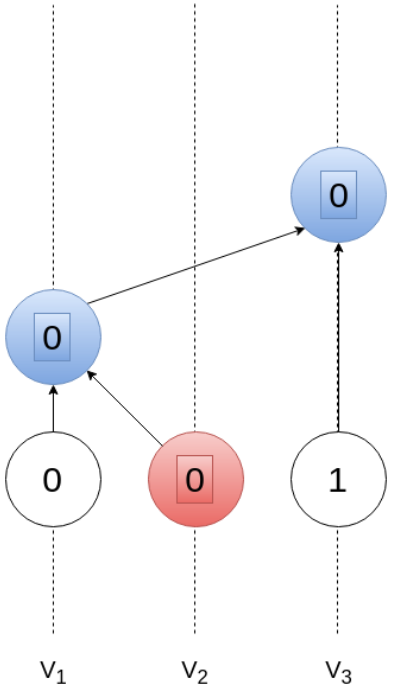
Has a changing set of validators.

Decides on the state of the EVM.

Safety is a threshold decision (how *much* safety?).

Is safe in asynchronous networks.

Is live in synchronous network.

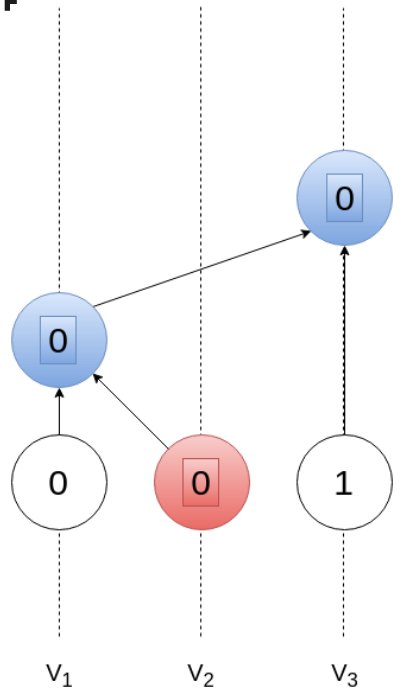


A roadmap for CbC Casper

Light client friendly Asynch BFT EVM Replication

With validator rotation

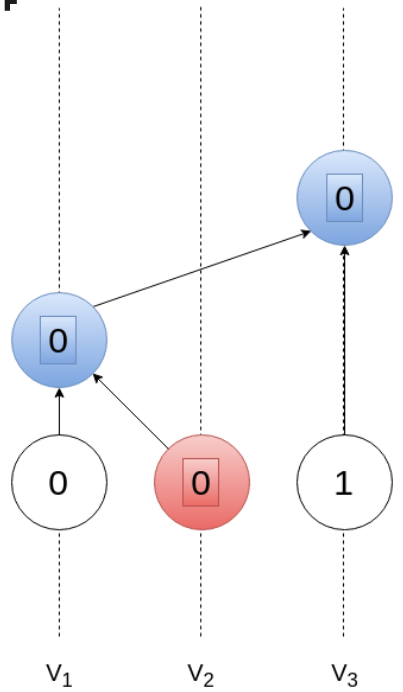
- Has a changing set of validators.
- Decides on the state of the EVM.
- Safety is a threshold decision (how *much* safety?).
- Is safe in asynchronous networks.
- Is live in synchronous network.
- **Is light client friendly.**



A roadmap for CbC Casper

Light client friendly Asynch BFT EVM Replication with VR for oligopolistic markets

- Has a changing set of validators.
- Decides on the state of the EVM.
- Safety is a threshold decision (how *much* safety?).
- Is safe in asynchronous networks.
- Is live in synchronous network.
- Is light client friendly.
- **Has fees, bonding + coalition-proof mechanism design.**



Q & A

Why is light client friendliness before mechanism design?
You and Vitalik have different roadmaps/visions! Isn't that bad?
When is Casper going to be ready?
When will the correct-by-construction Casper be ready?
Has any of this been implemented?
How are you going to make sure that the protocol is practical?
What are your favorite results in Casper research?
Why haven't you published a white paper yet?
Why aren't there more people working on this?
When is Casper going to be ready?
Are you going to take questions from the audience?
How do you really, truly feel about the DAO hard fork?

Conclusion

The correct-by-construction binary consensus has a trivial consensus safety proof because of the way that the specification is given.

We should be able to specify the whole Casper protocol using a correct-by-construction approach.

I am optimistic that it can be practical, and certain that it is educational.