Keywords

π-calculus, proof-of-stake, blockchain, types, Curry-Howard

ABSTRACT

We present Casper, a proof-of-stake protocol, and it's formal specification in π-calculus.

Formally introducing Casper

Vlad Zamfir
Ethereum
vldzmfr@gmail.com

L.G. Meredith
CSO, Synereo
greg@synereo.com

1. INTRODUCTION AND MOTIVATION

Consensus protocols build reliable processes out of unreliable processes e.g. a process (or protocol) that provides a service by replicating changes to a deterministic automata across many potentially unreliable or untrusted computers

Consensus isâĂę

CAP Theorem In the event of a network partition, a consensus protocol must choose between being available and being consistent

CAP Theorem Consistency: every correct server has an identical copy of the state

Availability: clients are able affect changes to the serverâĂŹs consensus state

Classes of Consensus Protocols Traditional consensus protocols tend to prefer consistency over availability in the event of a partition

Blockchain-based consensus protocols can prefer availability over consistency in the event of a partition

Exceptions that prove the rule Traditional consensus protocols that are available in the event of a partition tend to use a protocol extension to discover the validators in their partition, and then to proceed âĂŸnormallyâĂŹ with this subset of validators

On Chaining Blocks The only consensus-critical reason to chain blocks together is to restrict the number of possible futures/histories that clients must compare when they are coming to consensus

This is only necessary in consensus protocols that favour availability

On Chaining Blocks We will return to blockchain-based consensus protocols after reviewing traditional consensus protocols and economic consensus protocols.

1.0.1 Related work

1.0.2 Organization of the rest of the paper

2. ORGANIZING SOME CONSENSUS ON CONSENSUS

Traditional Consensus Protocols aim to have the following property: Correct nodes make a change to the consensus state if and only if they are certain that in a finite amount of time all other correct nodes will make the same change

Traditional Consensus Protocols They prove this property using the assumption that there exists no more than f (byzantine) faulty nodes, and that the remaining nodes are correct

Traditional Consensus Protocols If a correct node sees something X that causes it to change its state, it knows that all undecided correct nodes will also process that change, when they see X

Traditional Consensus Protocols If f nodes are faulty,

f + 1 node signatures are required for a node to believe that the message was signed by at least one correct node

Traditional Consensus Protocols âĂę. For consensus we must know that a majority of correct nodes approve of a change to the state f

Traditional Consensus Protocols Definition:

A âĂIJbyzantine quorumâĂİ is a set of nodes large enough such that a majority of correct nodes must be contained in that set. f

Traditional Consensus Protocols faulty nodes correct nodes majority of correct nodes byzantine quorum 0% 100% >50% >50% 33% 67% >33.5% >66.5% 50% 50% >25% >75% 80% 20% >10% >90% f

Traditional Consensus Protocols âĂę. The size of the byzantine quorum is

> (100% - f)/2 + f = 50% + f/2

âĂę. where f is the proportion of byzantine faults f

Traditional Consensus Protocols âĂę. Unfortunately, faulty nodes can prevent a byzantine quorum from forming âĂę. f

Traditional Consensus Protocols faulty nodes correct nodes majority of correct nodes byzantine quorum (c1 + c3) quorum surplus (c2 - c4) 0% 100% >50% >50% <50% 33% 67% >33.5% >66.5% <0.5% 50% 50% >25% >75% <-25% 80% 20% >10% >90% <-70% f

Traditional Consensus Protocols A fault tolerance f > âĚŞ would require a quorum size that is greater than âĚŤ. This quorum cannot be guaranteed to form if there are more than 1 - âĚŤ = âĚŞ faults,

âĂę.and we just assumed that f > âĚŞ. f

Traditional Consensus Protocols The highest fault tolerance we can hope for is: f = âĚŞ

|quorum| = 50% + (f + e)/2 = âĚŤ + e/2 |correct| = 100% - (f + e) = âĚŤ - e |correct| < |quorum| by 3e/2, and we can only tolerate faults if |correct| >= |quorum| f

Traditional Consensus Protocols A correct node that sees evidence of the acceptance of a change from a byzantine quorum knows that a majority of correct nodes have accepted that change

Traditional Consensus Protocols In this case it is possible for the correct node to safely commit the change to its state.

Traditional Consensus Protocols We need to assume that: A correct node âĂIJknowsâĂİ that other correct nodes will commit to the same change to the consensus because it knows that they are using the same logic to update their copy of the state

Economic Consensus Protocols In economic consensus, we relax the previous assumption.

The networkâĂŹs behaviour is constrained by the economics of the protocol rather than by the correctness of all-but-f nodes.

Economic Consensus Protocols.. Definition: Economic consensus protocols are consensus protocols that use economic mechanisms defined by the protocol to prevent byzantine faults.

Economic Consensus Protocols.. The security of an economic consensus protocol should be demonstrated through economic analysis. Economic analysis is unfortunately much harder to conduct than byzantine fault tolerance analysis.

Economic Consensus Protocols.. In general we will aim to show that an economic consensus protocol is: Incentive compatible: All nodes have a direct incentive to work together to update and secure the consensus against faults

Economic Consensus Protocols.. The most common type of incentive incompatibility is a majority coalitionâĂŹs incentive to censor the remaining minority to increase their profitability.

In the worst case, a majority coalition has an incentive to reverse history.

Economic Consensus Protocols.. In general we will aim to also show that an economic consensus protocol is: Robust against incentive attacks: It would require a large expenditure out of a large budget to break the protocolâĂŹs incentive compatibility.

Economic Consensus Protocols.. The most general incentive attack is called the bribe attack.

In this attack model, the attacker can directly bribe any node to deviate from its current behaviour. We assume that all (or almost all) nodes will deviate if it is profitable.

Economic Consensus Protocols.. Common assumption:

Economic consensus protocols enforce these fault-preventing incentive mechanisms only by specifying the behaviour of a ledger of digital assets defined in the consensus state

Economic Consensus Protocols.. This common assumption, together with the also common confusion of blockchain-based consensus protocols with economic consensus protocols is why people commonly refer to blockchains as âĂIJdistributed ledgersâĂİ

Economic Consensus Protocols.. Example:

Tendermint makes faults expensive by revoking the security deposits of faulty validators. These deposits must be placed by a validator for their blocks to represent valid changes to the consensus state.

Economic Consensus Protocols.. Counter-example:

Transactions-as-PoW is a consensus protocol that is expensive to attack, but which does not use an internal ledger to achieve its economics. It requires transactions to be computationally expensive and to refer to previous transactions.

Blockchain-based consensus protocols Achieve consensus by having nodes choose between competing histories (called blockchains or forks) based on a common fork-choice rule.

This is as opposed to achieving consensus exclusively by coordinating to extend history.

Blockchain-based consensus protocols The ability to choose one version of events after having adopted another is the defining characteristic of blockchains.

Chaining blocks reduces the number of possible version of histories.

Blockchain-based consensus protocols This property makes it possible for blockchains to favour availability over consistency in the event of a network partition.

Blockchain-based consesnsus: CA(P) Traditional consensus: CA(P)

Blockchain-based consensus protocols have a kind of fault-tolerance that doesnâĂŹt exist in the context of traditional protocols: The fork-based fault-tolerance is the proportion of network resources required to create a fork that is chosen by the fork-choice rule in favour of any fork that can be created by the remaining nodes.

Blockchain-based consensus protocols Traditionally, blockchain-based consensus protocols have a fork-based fault-tolerance of 50%.

Traditional fault tolerance refers to the number of nodes that can be byzantine while it is still impossible for correct nodes to disagree.

Blockchain-based consensus protocols Traditional fault tolerance: âĂIJHow many of the nodes who made this fork would need to be byzantine for it to possibly be reverted?âĂİ Fork-based fault tolerance: âĂIJHow many nodes need to be byzantine for the guarantee that the winning fork is one created by correct nodes?âĂİ

Decentralization analysis Traditional: âĂIJHow many nodes need to be byzantine for there to be a commitment that matches this one?âĂİ Fork-based: âĂIJHow many nodes need to be correct, to guarantee that the winning fork is the one created by correct nodes?âĂİ

Economic Consensus Protocols.. Example:

Bitcoin makes faults expensive by making blocks computationally expensive to produce, while rewarding miners through the issuance of new bitcoin only for producing blocks on the âĂIJheaviest chainâĂİ.

Economic Consensus Protocols.. Example, continued:

Miners who donâĂŹt produce blocks at the head of the heaviest chain have to incur the computational expense but are not compensated for itâĂę unless the blocks later become part of the heaviest chain.

Economic Consensus Protocols.. Example, continued:

By conducting a âĂIJ51% attackâĂİ, Bitcoin miners can work in a majority coalition to make a âĂIJlighterâĂİ fork into the âĂIJheaviestâĂİ fork.

Economic Consensus Protocols.. Example, continued:

We therefore commonly say that Bitcoin has a byzantine fault tolerance of 50%. Economic analysis, on the other hand, is not that simple..

Economic Consensus Protocols.. Example, continued: If we want to show that the Bitcoin protocol is secured by economics rather than by correct nodes, we must show that it is incentive compatible and robust to incentive attacks, ideally with as few assumptions as possible.

Economic Consensus Protocols..

Hence the term âĂIJdistributed ledgerâĂİ, which is unfortunately

3. CASPER, INFORMALLY

4. THE CALCULUS

One notable feature of the π-calculus is its ability to succinctly and faithfully model a number of phenomena of concurrent and distributed computing. Competition for resources amongst autonomously executing processes is a case in point. The expression

$$x?(y) \Rightarrow P \mid x!(u) \mid x?(v) \Rightarrow Q$$

is made by composing three processes, two of which, $x?(y) \Rightarrow P$ and $x?(v) \Rightarrow Q$ are seeking input from channel x before they launch their respective continuations, P and/or Q; while the third, $x!(u)$, is supplying output on that same said channel. Only one of the input-guarded processes will win, receiving u and binding it to the input variable, y, or respectively, v in the body of the corresponding continuation – while the loser remains in the input-guarded state awaiting input along channel x. The calculus is equinanimous, treating both outcomes as equally likely, and in this regard is unlike its sequential counterpart, the λ-calculus, in that it is not *confluent*. There is no guarantee that the different branches of computation must eventually converge. Note that just adding a new-scope around the expression

$$(\mathsf{new}\ x)(x?(y) \Rightarrow P \mid x!(u) \mid x?(v) \Rightarrow Q)$$

ensures that the competition is for a local resource, hidden from any external observer.

4.1 Our running process calculus

4.1.1 Syntax

$P ::= 0$	stopped process
$\mid x?(y_1, \ldots, y_n) \Rightarrow P$	input
$\mid x!(y_1, \ldots, y_n)$	output
$\mid (\mathsf{new}\ x)P$	new channel
$\mid P \mid Q$	parallel

Due to space limitations we do not treat replication, $!P$.

4.1.2 Free and bound names

$$\mathcal{FN}(0) := \emptyset$$
$$\mathcal{FN}(x?(y_1, \ldots, y_n) \Rightarrow P) := \{x\} \cup (\mathcal{FN}(P) \setminus \{y_1, \ldots y_n\})$$
$$\mathcal{FN}(x!(y_1, \ldots, y_n)) := \{x, y_1, \ldots, y_n\}$$
$$\mathcal{FN}((\mathsf{new}\ x)P) := \mathcal{FN}(P) \setminus \{x\}$$
$$\mathcal{FN}(P \mid Q) := \mathcal{FN}(P) \cup \mathcal{FN}(Q)$$

An occurrence of x in a process P is *bound* if it is not free. The set of names occurring in a process (bound or free) is denoted by $\mathcal{N}(P)$.

4.1.3 Structural congruence

The *structural congruence* of processes, noted $\equiv$, is the least congruence containing α-equivalence, $\equiv_\alpha$, making $(P, |, 0)$ into commutative monoids and satisfying

$$(\mathsf{new}\ x)(\mathsf{new}\ x)P \equiv (\mathsf{new}\ x)P$$

$$(\mathsf{new}\ x)(\mathsf{new}\ y)P \equiv (\mathsf{new}\ y)(\mathsf{new}\ x)P$$

$$((\mathsf{new}\ x)P) \mid Q \equiv (\mathsf{new}\ x)(P \mid Q)$$

4.1.4 Operational Semantics

$$\frac{|\vec{y}| = |\vec{z}|}{x?(\vec{y}) \Rightarrow P \mid x!(\vec{z}) \to P\{\vec{z}/\vec{y}\}} \quad (\textsc{Comm})$$

In addition, we have the following context rules:

$$\frac{P \to P'}{P \mid Q \to P' \mid Q} \quad (\textsc{Par})$$

$$\frac{P \to P'}{(\mathsf{new}\ x)P \to (\mathsf{new}\ x)P'} \quad (\textsc{New})$$

$$\frac{P \equiv P' \qquad P' \to Q' \qquad Q' \equiv Q}{P \to Q} \quad (\textsc{Equiv})$$

4.1.5 Bisimulation

Definition 4.1.1. *An* observation relation, $\downarrow$ *is the smallest relation satisfying the rules below.*

$$\frac{}{x!(\vec{y}) \downarrow x} \quad (\textsc{Out-barb})$$

$$\frac{P \downarrow x \ \mathit{or}\ Q \downarrow x}{P \mid Q \downarrow x} \quad (\textsc{Par-barb})$$

Notice that $x?(y) \Rightarrow P$ has no barb. Indeed, in π-calculus as well as other asynchronous calculi, an observer has no direct means to detect if a sent message has been received or not.

Definition 4.1.2. *An* barbed bisimulation, *is a symmetric binary relation $\mathcal{S}$ between agents such that $P\ \mathcal{S}\ Q$ implies:*

1. *If $P \to P'$ then $Q \to Q'$ and $P'\ \mathcal{S}\ Q'$.*
2. *If $P \downarrow x$, then $Q \downarrow x$.*

P is barbed bisimilar to Q, written $P \stackrel{\bullet}{\approx} Q$, if $P\ \mathcal{S}\ Q$ for some barbed bisimulation $\mathcal{S}$.

5. FORMALLY INTRODUCING CASPER

Acknowledgments. We would like to acknowledge Anthony D'Onofrio for putting us in touch.

6. APPENDIX: BECAUSE EVERY TECHNICAL PAPER NEEDS AN APPENDIX