The devil hides in the model: Reviewing Blockchain and BFT protocols

Antoine Durand[1] and Gérard Memmi[1]

LTCI, Télécom Paris, Institut polytechnique de Paris, France
firstname.lastname@telecom-paris.fr

Abstract. Recent advances in blockchains and Byzantine Fault Tolerant protocols have been numerous and varied in nature. However, making a fair and consistent comparison of existing protocols is a difficult task that must begin right at the execution model. In this work, we undergo a review of several prominent blockchain protocols including their models, *i.e.*, network synchrony, cryptographic assumptions, corruption, as well as latency and communication cost figures. This review illustrates the issues that can arise due to the lack of standardization on blockchain terminology. For example, we show that in two prominent blockchain protocols, seemingly minor technical details in the formulation leads to the execution model being strictly different from the intended one.

Keywords: Blockchain · Distributed Ledger Technology · Benchmark · Complexity

1 Introduction

In recent years, blockchains, and distributed ledger technologies in general, experienced a surge of interest, spawning an impressive number of research directions and concurrent works. This proliferation somehow contrasts with the widely recognized difficulty of designing and proving secure distributed cryptographic algorithms. Thorough evaluations of fault-tolerant Byzantine protocols such as blockchains are therefore becoming more complex. As a result, there is currently no recognized evaluation procedure for such protocols, which makes formal comparisons challenging; this also partially explains the lack of standardization and the increased difficulty of designing advanced blockchain applications, particularly when multiple protocols are involved.

In this paper, we propose an evaluation strategy that is centered around the execution model of each blockchain. We selected a collection of prominent and published blockchain protocols. Our contribution is that we interpreted their execution models through a common formulation supporting a fair comparison of their various assumptions, primitives, and, by extension, their performance. This approach allowed us define multiple common variables to carry out a detailed asymptotic evaluation of communication complexity and latency. As a result we are able to offer an assessment summary in Section 4 that is clear, rigorous, and comprehensive without being dependent on any one type of protocol.

We chose the following protocols that are representative of different techniques and working principles of blockchains, however, we could only consider sufficiently formalized protocols to enable a fair assessment of their model and properties. Nakamoto agreement [11] is certainly one of the most prominent, and we believe its analysis

applies to similar protocols such as Ethereum. We also chose Phantom [17] in order to have a second PoW-based protocol to compare with the Nakamoto agreement. Ouroboros Praos [4] is a non-PoW-based synchronous protocol, and Algorand [12] is somewhat similar to Ouroboros but in a partially synchronous setting. Tendermint [1] operates similarly to classical state machine replication protocols [5], and HoneyBadgerBFT [15] is asynchronous.

Existing benchmarks in published literature focus on qualitative comparisons and experimental performance evaluations [2]. In light of this, the closest work to ours is a taxonomy by Garay and Kiayias [10], for which we make a more enhanced contribution in this work with the inclusion of newer protocols, but most notably, the addition of a generic, multivariate performance analysis.

Outline. This paper is organized as follows. Section 2 is a concise introduction to the notations used to capture all of the protocol models. We analyze each protocol in Section 3, and discuss the resulting comparison in Section 4. Finally, we conclude in Section 5.

2 Model

To enable a generic discussion of the protocol's models and properties, we adopt the notations from Durand [7]. They are briefly summarized below.

Background The distributed system contains a set of nodes Π, as well as modules which serves as abstract representations of the components present in the system. A module m provides an interface to interact with nodes and also describes its behaviour through this interface; concretely, m is represented by its input and output domains and a validity predicate on sequences of inputs/outputs. Given a list of ideal primitives M, expressed as modules, the notation $P \in \mathcal{P}[M]$ means that P is a protocol that may interact with the modules in M. Such a protocol, P, is represented as a state machine run by each (honest) node and annotated with inputs/outputs. The set of executions of P that are conforming to its model are noted $\mathcal{M}_P^{\mathcal{A}}$, where $\mathcal{A}$ is a parameter that specifies the adversary's capabilities. $\mathcal{A}$ contains three components to indicate (1) whether the adversary is computationally bounded, (2) whether node corruption is dynamic, weakly dynamic, or static, and (3) the corruption structure [13] that must be respected. More specifically, a corruption structure $\mathcal{C}$ contains all sets of nodes that may be corrupted. This generalization of corruption thresholds is well suited for models that leverage Proof-of-X for Sybil resistance.

The notation $\mathcal{M}_P^{\mathcal{A}} \models^{\lambda} S$ indicates that protocol P satisfies specification S with probability $1 - O(2^{-\lambda})$, where λ is the security parameter, and the specification S is also represented by a module.

Specification The selected protocols slightly vary in the formulation of their specification. Thus, we chose Atomic Broadcast (ABC) [6] as a common specification based on the state machine replication paradigm. An ABC protocol is a protocol where nodes may input and output values from a message set M, such that,

- *ABC-Consistency:* For any honest nodes p_1 and p_2, for any $k \in \mathbb{N}$ such that p_1 and p_2 both made at least k outputs, then p_1 and p_2' k-th outputs equal.
- *ABC-Liveness:* All honest players' inputs are eventually output by all honest nodes.

Primitives All models include a point-to-point network module, however there are three possible variants depending on synchronicity level, namely:

- "sync_net": Any sent message m takes at most δ time to be received, and $\Delta \geq \delta$ is given as a parameter to the nodes.
- "weak_net": Any sent message m takes at most δ time to be received, but Δ is unknown to the nodes.
- "async_net": There is no bound on the message delays.

The models include a few cryptographic primitives, also modeled as modules. We do not give explicit module definitions here and refer instead to standard cryptographic definitions [14]. These are: random oracle "RO", digital signatures "signatures" (including a public key infrastructure), public parameters "params", and Verifiable Random Functions "VRF".

The performance evaluation is a multivariate asymptotic evaluation of communication cost and latency. Communication cost ($\mathcal{CC}$) is the number of bits sent by honest nodes, and latency ($\mathcal{L}$) is the time elapsed between submission of a message an its delivery by all honest nodes. The common input variables of the asymptotic analysis are:

- Δ is the known upper bound on network delays, for "sync_net" networks.
- δ is the unknown upper bound on network delays, for "weak_net" networks.
- n is the number of nodes.
- b is the amortized size of the protocol output payload.
- $\text{poly}_v(\lambda)$ is a function polynomial in the security parameter λ, and arbitrary in v.

It's important to note that the above definitions only require the presence of an ABC protocol and a network. Thus these variables can be evaluated and analyzed for any blockchain protocol.

To simultaneously model corruption for Proof-of-Work (PoW), Proof-of-Stake (PoS), and classical BFT protocols, we use weighted corruption thresholds. Given the nodes' weights $W : \Pi \mapsto [0,1]$ such that $\sum_{p\in\Pi} W(p) = 1$ and a threshold $t \in [0,1]$; then the adversary respects the corruption structure $\mathcal{C}_{W,t}$ iff at all times during the execution, the set of corrupt nodes $C \subseteq \Pi$ is such that $\sum_{p\in C} W(p) < t$.

3 Protocol Analysis

3.1 Using the Bitcoin backbone protocol

For Nakamoto agreement[1] and Ouroboros Praos, we rely on the backbone protocol formalism from Garay *et al.* [11,4]. They present two analyses, one with a synchronous network and one extended to a partially synchronous network, the latter being similarly defined in Ouroboros Praos [4]. They describe a network offering a global round clock to all nodes. Then, sent messages are delivered after up to Δ rounds, meaning that such rounds do not have a prescribed duration and act more as "real time step" or "time slot".

However, we believe there is a slight issue in the formulation used. First of all, it's important to note that both protocols are stated to be secure assuming that less than *half*

[1] We use the term "Nakamoto agreement" or "Nakamoto" for short to refer to Bitcoin's underlying *ABC* protocol.

of the stake/hashpower is owned by malicious nodes. This is surprising since consensus cannot be solved in partially synchronous networks with more than a third of the honest nodes. On the other hand, if there is an honest majority, a strongly synchronous network is required to solve consensus.

Moreover, the only difference between partial and strong synchrony is that the Δ bound is made available to nodes in the strongly synchronous case. Therefore, with careful examination, we observe that both Nakamoto and Ouroboros Praos actually requires the value Δ to be known to execute correctly, and as a result they could be better described as synchronous protocols. Essentially, for both cases, the protocols can be proven secure assuming they are fed with the correct parameters, but if nodes are able to output transactions with a known probability of error, *i.e.*, if they *assume* the protocol is secure, then they are also able to compute Δ.

For Ouroboros Praos, the protocol takes a parameter f that tunes the probability for a node to be eligible to multicast the current block. To know whether a given node is eligible to multicast a block, a procedure taking f as a parameter is executed *within the protocol*. Then, to prove the security of the protocol, the authors require that f satisfies an inequality [4, Theorem 9 equation 12], which encodes the "Majority of Honest Stake" assumption but also depends on Δ. Notably, if f and all the other protocol parameters are known to the nodes, then assuming that the inequality holds, this implies that nodes can solve it to obtain (an upper bound on) Δ.

We adopt a similar reasoning with Nakamoto. Here, Δ is related to other known parameters through an inequality [11, Honest Majority Assumption (Bounded Delay)], which must hold to prove the security of the protocol. Therefore, nodes can solve it to compute Δ. Formally, these remarks can be expressed with the following theorem.

Theorem 1 (Nakamoto and Praos are strongly synchronous).

$$\exists P \in \mathcal{P}[M_{\text{Praos}}],\ \mathcal{M}_P^{\mathcal{A}_{\text{Praos}}} \models \text{strong\_net}$$
$$\exists P' \in \mathcal{P}[M_{\text{Nakamoto}}],\ \mathcal{M}_{P'}^{\mathcal{A}_{\text{Nakamoto}}} \models \text{strong\_net}$$

Where $(M_{\text{Nakamoto}}, \mathcal{A}_{\text{Nakamoto}})$ *and* $(M_{\text{Praos}}, \mathcal{A}_{\text{Praos}})$ *are the models of Nakamoto agreement and Ouroboros Praos, respectively.*

Proof. To implement the strong_net module, P and P' work similarly. Initially, nodes compute a bound $D \geq \Delta$ using the procedure described above. Upon request of the Δ value through the interface of strong_net, the value D is returned. Any other interaction is proxied to the underlying weak_net module. Knowing that a weakly synchronous network with a known bound on δ is exactly a synchronous network; therefore, the conclusion is reached.

3.2 Nakamoto agreement

Model. We have already shown that the model from Garay *et al.* [11] is strongly synchronous, with a computational adversary. Their model is completed with a Random Oracle, which is modified to integrate the Proof-of-Work mining primitive. That is to say, the Random Oracle is given the ability to answer "mining" queries from the nodes, with a limit of q queries per node per network round (or 1 in the "partially" synchronous

analysis). Then, an additional interface is added to be able to verify the result of a query without having to make the same query again and be limited by q.

Regarding the corruption structure, they state an "Honest Majority Assumption" such that the proportion of hashpower (expressed in RO queries per round) available to the malicious nodes is lower than $\frac{1-d(\lambda,\Delta)}{2-d(\lambda,\Delta)}$ with d a function bounded between 0 and 1. This "Honest Majority Assumption" is easily modeled as a weighted corruption structure with each node being weighted by its hashpower. As expected, d is an increasing function, meaning that the amount of tolerated malicious hashpower gets further from the optimal value $(1/2)$ down to 0 as the security parameter is increased. Node corruptions take effect immediately, therefore the adversary is dynamic.

Metrics. In the Bitcoin Backbone analysis, Garay *et al.* articulate their proofs on the assumption of a *typical execution* that roughly states that parties produce blocks at a rate close enough to their expected value (*i.e.*, hashpower). Then, they show that any execution of k rounds is typical with probability $1-e^{-\Theta_\varepsilon(k)}$, where ε is a variable that quantifies how close the block production rate is to its expected value. In turn, the proofs will rely on ε being appropriately bounded, an assumption that is integrated in their version of "honest majority assumption". In particular, this assumption gives a bound on ε that depends on Δ, hence, an execution which is longer than $\text{poly}_\Delta(\lambda)$ rounds is typical with overwhelming probability.

Assuming a typical execution, Garay *et al.* prove that it takes $\frac{4k}{1-\varepsilon}$ rounds for a transaction to be confirmed by k blocks. Thus, if $k = \text{poly}_(\lambda)$ the latency of transaction confirmation is $\mathcal{L}(\text{NAKAMOTO}) = \text{poly}_\Delta(\lambda)$.

The analysis regarding communication complexity is simpler: All b bits from the output are blocks that have been multicast once, hence the communication cost is at least bn. Furthermore, this cost is only increased if the adversary forces orphaned blocks. Since all messages on the network are blocks with a valid PoW, it is clear that the overall number of blocks received by honest nodes is a constant factor of the number of blocks that will end up in the blockchain. Hence, $\mathcal{CC}(\text{NAKAMOTO}) = \Theta(bn)$.

3.3 Ouroboros Praos

Model. We have already shown that the model from Bernardo *et al.* [4] is strongly synchronous, with Byzantine faults and a computational adversary. Like with the Nakamoto agreement, we now complete the model.

The existence of the Random Oracle and digital signatures is assumed. Node corruptions take immediate effect and the adversary is fully dynamic. This is possible due to the usage of VRFs, on a principle similar to how the Bitcoin miner can be corrupted just after sending a block without issue. Ouroboros nodes use the VRF to learn locally whether they are randomly selected to multicast a block, and once the block is sent they no longer have a privileged role. The verifiability of the VRF output ensures that nodes cannot cheat their eligibility.

Regarding the corruption structure, Bernardo *et al.* assume that the proportion of stake owned by honest nodes is higher than $\frac{1}{2}d(\lambda,\Delta)$, where d is an (increasing) function lower bounded by 1. This assumption is modeled by a weighted corruption structure using stake as weights.

Metrics. In Ouroboros Praos, for every time slot, each node can be independently elected to multicast a new block, and the probability to be elected is only a function of the amount of the owned stake. Generating the randomness to seed the election of block leaders is implemented by having each leader include a VRF output in their block, and periodically concatenating the VRF output to form a random seed for the future VRF evaluation.

Given this method of block production, Ouroboros's properties can be proven using only combinatorial arguments on the distribution of honest and malicious leaders in the block tree. This is done through the analysis of "characteristics strings", which are an encoding of the schedule of honest and malicious leaders. Bernardo *et al.* show that a string of length of k time slots is "forkable" with probability $\text{negl}(k)$, for an appropriate notion of "forkable" that is later used to prove the protocol security. In particular, the distribution of characteristics strings only depends on the (stakewise) proportion of malicious leaders, and therefore k is only a function of the security parameter, *i.e.* $k = \text{poly}(\lambda)$ and $\mathcal{L}(\text{PRAOS}) = \Theta(\Delta k)$

3.4 Tendermint

Our discussion of Tendermint is based on an analysis by Amoussou-Guenou *et al.* [1].

Model. Tendermint follows more classical approaches to State Machine Replication [5] and as such fits very nicely into the ABC formulation. Amoussou-Guenou *et al.* assume a partially synchronous network with a formalism based on a Global Stabilization Time (GST), which is equivalent to having an unknown network delay δ. All nodes sign their messages with a digital signature algorithm. Tendermint uses a hash function that can be modeled with a Random Oracle. The corruption threshold is $\lfloor n/3 \rfloor$ nodes, which is simply represented as a corruption structure with equal weights for all nodes.The adversarial adaptivity is not explicitly specified, however it is straightforward to see that it can be dynamic; the leader is not expected to be honest, and the schedule of leaders may as be well known when the adversary choose corruptions.

Metrics. Tendermint's normal case operation is reminiscent of PBFT [5], except that leaders are always changed after each broadcast, whether they be successful or not. For each block, there is a leader that will execute a reliable broadcast protocol implemented through two all-to-all voting rounds. To optimize the bandwidth, only the hash of the block is included in the voting messages. This requires the b bytes of the block content to be sent to all n nodes, and the additional cost of n^2 bytes for the reliable broadcast, taking three communication steps. That is, a single leader attempting to append a block takes $\Theta(bn + n^2)$ bit complexity and $\Theta(1)$ latency.

Malicious leaders can ensure that their attempt does not succeed, and because the leader order is arbitrary, all malicious nodes may be leaders first, thus increasing latency by a factor $\max_{C \in \mathcal{C}} \#C = \Theta(n)$. This does not impact the overall communication cost however, because all honest leaders will also append a block and only a constant fraction of nodes are malicious. Tendermint is optimistic in the sense that, in failure free executions, transactions are immediately appended and the latency becomes $\Theta(\delta)$.

3.5 HoneyBadgerBFT

Model. In HoneyBadger BFT [15](HBBFT), the network is asynchronous, up to $1/3$ of the nodes may be corrupt, and the adversary is static. The protocol makes use of a hash

function and digital signatures. Miller *et al.* explicitly assume a "Purely asynchronous network", "Static Byzantine faults" and "Trusted setup", and they aim to solve Atomic Broadcast. The setup only serves to implement a PKI for the digital signatures and a common coin subprotocol.

Metrics. HoneyBadgerBFT uses the reduction to a reliable broadcast and a binary consensus from Ben-Or *et al.* [3] to implement a variant of the consensus problem, namely an asynchronous common subset.

The reliable broadcast protocol terminates in three rounds and has a bit complexity of $\Theta(bn + n^2 \log(n) \operatorname{poly}(\lambda))$. The binary consensus is an asynchronous probabilistic protocol with $\Theta(n^2 \operatorname{poly}(\lambda))$ bit complexity. At each round it has $1/2$ probability to terminate. HBBFT must wait for all their consensus instances to terminate, and the time required is $\Theta(\log(n))$ rounds on average.

However, one of the achievements of HBBFT is that at the end of this procedure, it commits data from all node inputs. That is, if all the nodes have an input of size B, then the batch committed will be of size $\Theta(nB)$ bits. Hence, for b bits committed, there are $\Theta(bn + n^3 \log(n) \operatorname{poly}(\lambda))$ bits received by honest players.

Miller *et al.* do provide a complexity analysis, but they state their results in terms of *overhead*, *i.e.*, the total cost divided by b. Furthermore, by the specifying minimum input size (batching policy) of $\Omega(n^2 \log(n) \operatorname{poly}(\lambda)) = O(b)$, they obtain an $O(n)$ figure which is a constant per-node overhead, which essentially amounts to looking at the communication complexity *assuming the block contents is the dominating cost*. This emphasis on the low overhead is less visible in our results, although it is translated by the fact that HBBFT complexity on the b factor is bn instead of bn^2 for other BFT-style protocols, *i.e.* a reduction factor of n is consistent with the authors' analysis.

3.6 Phantom

Model. Phantom [17] is a Proof-of-Work protocol whose mining operation is very similar to Nakamoto, except that blocks may have more than one parent. As a result, the model is essentially the same as for Nakamoto, however Sompolinsky and Zohar use their own formulation based on an earlier proposal [16]. Phantom's security statement (Theorem 4) merges the ABC-Consistency and ABC-Liveness properties into a single one, although its formulation is specifically tied to the blockchain structure. Additionally, because the authors explicitly aim for a generalized version of Nakamoto, we are confident in choosing ABC to faithfully capture Phantom's properties.

The network is strongly synchronous; the authors state "if an honest node $v \in \mathcal{N}$ sends a message of size b MB at time t, it arrives at all honest nodes by time $t + D$ the latest.", where a bound one D is known to the protocol. Up to $\frac{1}{2}(1 - d(\lambda, \Delta))$ of the hashpower may be corrupt, with d an (increasing) function lower bounded by 0. Although not mentioned explicitly, the adversary is dynamic for the same reason as with Nakamoto agreement.

The Proof-of-Work mining is modelled by a Poisson process. Although common modeling of the mining process results in a Binomial distribution, this is not an issue since this distribution converges towards the Poisson distribution whenever the number of tries goes to infinity. As with Nakamoto, we also require the Random Oracle to answer hash queries unrelated to mining.

Metrics. In Phantom, the block structure is a Directed Acyclic Graph (DAG) which allows blocks to be linked to any number of the previous blocks instead of one. A key technique of Phantom is that PoW merely serves as a network-level synchronization primitive, thus decoupling the mining hardness from the protocol security. And indeed the mining rate is independent from λ and only required to be high enough so that blocks are not mined faster than they can be transmitted through the network. As such, there is little restriction on the mining process. More precisely, the entirety of the online protocol consists of mining on top of all the blocks with no successor in the block graph.

The online protocol maintains the DAG that grows over time, and another sub-protocol independently determines which values from the DAG will be the ABC output.

Phantom implements a procedure *Risk* that provides a bound on the probability that safety will not hold for this transaction. The authors then show that the bound returned is smaller than a given ε after $O(\log(\frac{1}{\varepsilon}))$ honest blocks are created. Hence, with $\varepsilon = O(e^{-\lambda})$, and since this procedure requires the upper bound Δ, we have that the number of honest blocks required is $\text{poly}_{\Delta}(\lambda)$. The time taken for this to happen is obtained by multiplying it by the rate of honest block production, which only depends on Δ. Then, in order for all the nodes to be aware of these blocks, an additional Δ overhead is added, resulting in $O(\Delta + \text{poly}_{\Delta}(\lambda))$ latency.

On the other hand, communication complexity is simpler. In fact, the same analysis as the other blockchain style algorithms is applicable. Blocks all require a valid PoW to be sent through the network, and the PoW creation rate is bounded according to the corruption structure, thus reaching an optimal $\mathcal{CC}(\text{PHANTOM}) = \Theta(bn)$.

3.7 Algorand

Model. Algorand [12] uses a partially synchronous network, the Random Oracle, forward secure signatures, and "VRFs". As expected in a partially synchronous network, a proportion $h > \frac{2}{3}$ of the stake is assumed to be owned by honest nodes. However, since Algorand uses an election mechanism, the protocol failure probability increases as h goes to $\frac{2}{3}$. Like with Ouroboros Praos, their usage of VRFs gives the possibility to tolerate dynamic adversaries.

Metrics. Like Ouroboros Praos, Algorand relies on a public randomness computed in previous blocks. It is used to elect a committee (instead of a leader) *at each round* that will have sufficiently many honest nodes, with overwhelming probability. These committees run a consensus protocol, which does not require a private state from the nodes (except from their private key), since committees would not be able to pass it on to the next committee. Except from this property, common techniques from BFT algorithms can be used to reach agreement in a constant number of rounds. In particular, since it has been ensured that committees have a constant fraction of honest nodes, standard quorum-based arguments are still valid. More precisely, at each round, each user has a fixed probability p to be part of the committee. To bound the number of malicious nodes in a committee, the authors leverage the fact that $\forall t' < t$; the probability of having at most t' malicious nodes in a uniformly sampled committee of size k is $1 - \text{negl}(k)$. Thus, the committees expected size is $\text{poly}(\lambda)$.

As a result, each round that would be equivalent to a n^2 all-to-all communication in a traditional BFT algorithm is now a "committee-to-all", $\Theta(n\,\text{poly}(\lambda))$ communication.

Additionally, Algorand optimizes bandwidth through a block proposing step similar to a leader election. The average committee size for that step is the smallest such that there is at least one proposer with overwhelming probability, *i.e.*, $\text{poly}(\lambda)$ asymptotically.

4 Discussion

Table 1 summarizes the above results. Concretely, For each protocol P in Table 1, we have made the following claim.

$$P \in \mathcal{P}[\text{params}, \text{RO}, M],\ \mathcal{M}_P^{(\text{computational}, \mathcal{C}_{(W,\alpha)}, (E,i,p)\mapsto\top)} \models^{\lambda} ABC \tag{1}$$

where: $W(p) := h_p(\sum_{q\in\Pi} h_q)^{-1}$ if the protocol uses the PoW module, or, if the protocol has a *stake* public parameter, $W(p)$ is p's relative stake. α is the contents of the "Adversary" column, M is the contents of the "Modules" column. We also claimed that $\mathcal{L}(P)$ is equal to the contents of the "Latency" column and that $\mathcal{CC}(P)$ is equal to the contents of the "Communication Cost" column.

Table 1: Comparison Summary.

Algorithm	Corruption threshold	Modules	Latency	Communication Cost
Nakamoto [11]	$1/2-\varepsilon(\lambda,\Delta)$	sync_net, PoW	$\text{poly}_\Delta(\lambda)$	$\Theta(bn)$
Phantom [17]	$1/2-\varepsilon(\lambda,\Delta)$	sync_net, PoW	$\Theta(\Delta+\text{poly}_\Delta(\lambda))$	$\Theta(bn)$
Ouroboros Praos [4]	$1/2-\varepsilon(\lambda,\Delta)$	sync_net, signatures, VRF	$\Theta(\Delta\,\text{poly}(\lambda))$	$\Theta(bn)$
Algorand [12]	$1/3-\varepsilon(\lambda)$	weak_net, signatures, VRF	$\Theta(\delta)$	$\Theta(bn\,\text{poly}(\lambda))$
Tendermint [1]	$1/3$	weak_net, signatures	$\Theta(\delta n)$, f.f. $\Theta(\delta)$	$\Theta(bn+n^2)$
HoneyBadgerBFT [15]	$1/3$	async_net, signatures	$\Theta(\delta\log(n))$	$\Theta(bn+n^3\log(n)\,\text{poly}(\lambda))$

This table clearly outlines the relation between the protocol's models and their performance metrics. The major drawback of Bitcoin is that its latency has an arbitrary dependence on Δ, which could very well be exponential. The same concern also applies to Phantom, and this is due to the fact that the confidence in a transaction depends on the time it takes for a PoW string to be transmitted. Moreover, it's worth remembering that Δ must hold for all messages, *of any length*. Therefore, increasing the block size may change Δ and results in a theoretically unknown impact on the protocol latency. These considerations point towards analyzing how latency depends on Δ and would be a great step forward in the assessment of PoW-based protocols.

Interestingly, we can see that all three synchronous protocols have their corruption threshold dependent on Δ, and for the same reason for each of them, the "honest majority" assumption is stated as an inequality that depends on Δ. As expected, the $1/2$ corruption threshold is synonymous with strong synchrony, and the PoW oracle is able to replace digital signatures. The presence of "$-\varepsilon(...)$" in the corruption threshold is indicative of protocols making use of the election mechanism, which also imply a factor $\text{poly}(\lambda)$ in its costs. Algorand is interesting in this regard, as it uses a large elected committee instead of leaders, thus removing the Δ factor in the threshold reduction ε, but gaining the $\text{poly}(\lambda)$ in communication cost. Reducing this communication cost by a constant factor is possible, *e.g.*, through sharding [8,9]. On the other hand Tendermint and HoneyBadger BFT doesn't use the election mechanism and bear a communication cost at least quadratic in n.

5 Conclusion

We presented a brief review of a selection of prominent published blockchain protocols. This review aimed at putting the protocol execution models on a common footing and at analyzing and comparing communication cost and latency as a function of multiple common variables. Our work puts an emphasis on having a streamlined terminology that fits the wide range of executions models. We were able to highlight and solve confusions and non-trivial issues that arise due to the lack of standardized formal definitions for common distributed primitives. We believe that this work can serve as a basis for facilitating the comparison of blockchain protocols as well as alleviating both the difficulties of standardization and fine-grained interoperability.

References

1. Amoussou-Guenou, Y., Pozzo, A.D., Potop-Butucaru, M., Tucci Piergiovanni, S.: Dissecting tendermint. In: NETYS 2019, June 19-21, Marrakech, Morocco
2. Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., Danezis, G.: Sok: Consensus in the age of blockchains. In: AFT 2019, Zurich, Switzerland
3. Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience (extended abstract). In: PODC, Los Angeles, California, USA, August 14-17, 1994
4. Bernardo, D., Gaži, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: EUROCRYPT (2018)
5. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (Nov 2002)
6. Cristian, F., Aghili, H., Strong, H.R., Dolev, D.: Atomic broadcast: From simple message diffusion to byzantine agreement. Inf. Comput. **118**(1), 158–179 (1995)
7. Durand, A.: Byzantine consensus and blockchain : Models unification and new protocols. Theses, Polytechnic Institute of Paris (Nov 2021)
8. Durand, A., Anceaume, E., Ludinard, R.: Stakecube: Combining sharding and proof-of-stake to build fork-free secure permissionless distributed ledgers. In: NETYS 2019, Marrakech, Morocco, June 19-21, 2019. LNCS, vol. 11704, pp. 148–165. Springer (2019)
9. Durand, A., Hébert, G., Toumi, K., Memmi, G., Anceaume, E.: The stakecube blockchain : Instantiation, evaluation amp; applications. In: 2020 Second International Conference on Blockchain Computing and Applications (BCCA). pp. 9–15 (2020)
10. Garay, J.A., Kiayias, A.: Sok: A consensus taxonomy in the blockchain era. In: CT-RSA 2020, San Francisco, CA, USA, February 24-28. LNCS
11. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: EUROCRYPT 2015, Sofia, Bulgaria, April 26-30, 2015. Updated version on IACR Cryptol. ePrint Arch. **2014**, 765.
12. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. In: SOSP 2017, Shanghai, China, October 28-31, 2017
13. Hirt, M., Maurer, U.M.: Complete characterization of adversaries tolerable in secure multi-party computation. In: PODC '97, Santa Barbara, California, USA, August 21-24. ACM
14. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. CRC Press (2014)
15. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The honey badger of BFT protocols. In: ACM SIGSAC 2016, Vienna, Austria, October 24-28, 2016
16. Sompolinsky, Y., Lewenberg, Y., Zohar, A.: SPECTRE: A fast and scalable cryptocurrency protocol. IACR Cryptol. ePrint Arch. **2016**, 1159
17. Sompolinsky, Y., Zohar, A.: PHANTOM and GHOSTDAG: A scalable generalization of nakamoto consensus. IACR Cryptol. ePrint Arch. **2018**, 104