Brief Announcement: Holistic Verification of Blockchain Consensus

Nathalie Bertrand, Vincent Gramoli, Igor Konnov, Marijana Lazić, Josef Widder, Pierre Tholoniat

Brief Announcement: Holistic Verification of Blockchain Consensus

Nathalie Bertrand
nathalie.bertrand@inria.fr
Univ Rennes, Inria, CNRS, IRISA
Rennes, France

Vincent Gramoli
vincent.gramoli@sydney.edu.au
University of Sydney
Sydney, Australia

Igor Konnov
igor@informal.systems
Informal Systems
Wien, Austria

Marijana Lazić
lazic@in.tum.de
TU Munich
Munich, Germany

Pierre Tholoniat
pierre@cs.columbia.edu
Columbia University
New York City, USA

Josef Widder
josef@informal.systems
Informal Systems
Wien, Austria

ACM Reference Format:
Nathalie Bertrand, Vincent Gramoli, Igor Konnov, Marijana Lazić, Pierre Tholoniat, and Josef Widder. 2022. Brief Announcement: Holistic Verification of Blockchain Consensus. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing (PODC '22), July 25–29, 2022, Salerno, Italy*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3519270.3538468

1 INTRODUCTION AND PRELIMINARIES

Motivation. Today, the market capitalization of the seminal blockchain, Bitcoin, is about $803B which incentivizes malicious participants to find problematic executions that would allow them to steal financial assets. As the blockchain requires a distributed set of machines to agree on a unique block of transactions to be appended to the chain, attackers naturally try to exploit consensus vulnerabilities to double spend. As a result, formally verifying that a blockchain consensus protocol is safe and live is key to mitigate financial losses. Recent progress in mechanical proofs represent the first steps towards verifying blockchain consensus. The parameterized model checking of threshold automata (TAs) has recently proved instrumental in verifying fully asynchronous parts of consensus algorithms, like broadcast algorithms [4]. The aforementioned reduction technique cannot apply to partial synchrony: moving the message reception step to a later point in the execution might violate an assumed message delay.

Contribution. In this paper, we verify holistically the safety and liveness properties of the DBFT [2] Byzantine consensus used in the Red Belly Blockchain system [3], a scalable blockchain used in production. Our approach is holistic because it starts from the pseudocode of the distributed algorithm as typically presented in the distributed computing literature, models this pseudocode and its components into disambiguated TA, model checks the desired properties of these components expressed in LTL formulae, simplifies the TA of the consensus algorithm with these verified properties and model checks the safety and liveness of the consensus protocol. The advantage is that the formally verified algorithm matches the pseudocode and no user-defined invariants or proofs need to be checked, which drastically reduces the risks of human errors.

Model. We consider n asynchronous sequential processes $\Pi = \{p_1, \ldots, p_n\}$ that communicate by exchanging messages through an asynchronous reliable fully connected point-to-point network. A set of $f \leq t < n/3$ processes are *Byzantine*, the rest are *correct*. Distributed algorithms are communication-closed and their execution consists in an interleaving of the individual steps taken by the processes. A *threshold automaton* (TA) describes the behavior of a process in a distributed algorithm. Its nodes represent local states, and labeled edges are *guarded rules*. In order to verify the safety and liveness properties of our distributed algorithms on a multi-round TA, we express these properties in fragments of linear temporal logic (LTL). Since the algorithms are communication-closed, the verification on the multi-round TAs reduces to one-round TAs. We then use the ByMC model checker to automatically verify these fragments.

2 THE BINARY VALUE BROADCAST

Algorithm 1 Binary value broadcast for process p_i.

```
1: bv-broadcast(BV, ⟨val, i⟩):
2:   broadcast(BV, ⟨val, i⟩)
3:   repeat:
4:     if (BV, ⟨v, ∗⟩) received from (t + 1) distinct proc. (not yet re-bcast) then
5:       broadcast(BV, ⟨v, i⟩)
6:     if (BV, ⟨v, ∗⟩) received from (2t + 1) distinct processes then
7:       contestants ← contestants ∪ {v}
```

Our holistic verification approach consists of decomposing a distributed algorithm into encapsulated components of pseudocode that can be modelled in TAs and verified in isolation to obtain a simplified TA that is amenable to automated verification. The *binary value broadcast* (bv-broadcast) [5] is a communication primitive guaranteeing that all binary values "bv-delivered" were "bv-broadcast" by a correct process. Each process starts this algorithm in one of two states, depending on its input value 0 or 1. Once a correct process receives a value from t+1 distinct processes, it broadcasts it if it did not do it already (line 4); broadcast is not

PODC '22, July 25–29, 2022, Salerno, Italy.

ACM ISBN 978-1-4503-9262-4/22/07
https://doi.org/10.1145/3519270.3538468

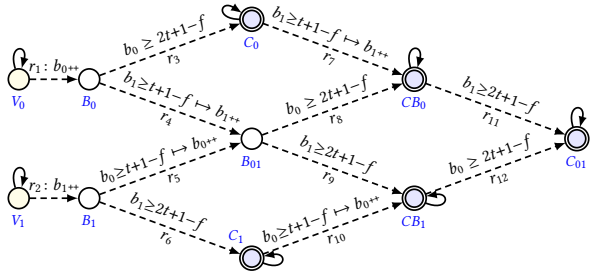

Figure 1: The threshold automaton model for the binary value broadcast. The locations of the automaton correspond to the exclusive situations for a correct process. The initial location V_0 (resp. V_1) indicate that the process initially has value 0 (resp. 1). Processes in B_0 (resp. B_1, B_{01}) have broadcast 0 (resp. 1, {0, 1}) without delivering anything. Processes in CB_0 (resp. CB_1) have broadcast {0, 1} and delivered 0 (resp. 1). Processes in C_0 (resp. C_1, C_{01}) have both broadcast and delivered 0 (resp. 1, {0, 1}).

Byzantine fault tolerant and just sends a message to all the other processes. Once a correct process receives a value from $2t+1$ distinct processes, it "bv-delivers" it. Here such a bv-delivery at process p_i is modeled by adding the value to the set *contestants*, which will simplify the pseudocode of the Byzantine consensus algorithm in Section 3.

From pseudocode to TA. To model the bv-broadcast pseudocode (Alg. 1) parameterized by n and t into a TA (Fig. 1), we define b_0 and b_1 as the global variables counting the number of the two types of messages (BV, $\langle 0, i\rangle$) and (BV, $\langle 1, i\rangle$) sent by correct processes. For example, $b_{0}{}^{++}$ models a process broadcasting message (BV, $\langle 0, i\rangle$). Because the algorithm only counts messages regardless of sender identities, we replace the messages from the pseudocode into b_0 and b_1 shared variables that are increased whenever a message is sent. To model that, among the received messages, f of them could have been sent by Byzantine processes, we map the 'if' statement of Alg. 1, comparing $t+1$ to the number of receptions from distinct processes, to the TA guards, comparing the number b_1+f of messages sent to $t+1$. As b_1 counts the messages *sent* by correct processes and f is the number of faulty processes that can send arbitrary values, a correct process can move from B_0 to B_{01} as soon as $t+1-f$ correct processes have sent 1. As a result, the guard of rule r_4 only evaluates over global send variables as: if more than $t+1$ messages of type b_1 have been *sent* by correct processes (hence the guard $b_1 \geq t+1-f$), then the shared variable b_1 is incremented, mimicking the broadcast of a new message of type b_1. Rule r_3 corresponds to lines 6–7 and delivers value $v = 0$ by storing it into variable *contestants* upon reception of this value from $2t+1$ distinct processes. Hence, reaching location C_0 in the TA indicates that the value 0 has been delivered. As a process might stay in this location forever, we add a self-loop with guard condition set to true.

Properties. As was previously proved by hand [5], the bv-broadcast primitive satisfies four properties: BV-Justification, BV-Obligation, BV-Uniformity and BV-Termination. Here, we formalize these properties in linear temporal logic (LTL) to formally and automatically prove they hold. We verify them for any parameters n and $t < n/3$ in less then 10 seconds.

A fairness assumption to solve consensus. The traditional approach to establishing guaranteed liveness properties in verification is to require that all fair executions, instead of all executions, satisfy the properties. We thus introduce the fairness assumption that will be crucial in the rest of this paper. In order to define it, we first define a *good execution* of the bv-broadcast with respect to binary value v as an execution:

Definition 2.1 (v-good bv-broadcast*).* A bv-broadcast execution is *v-good* if all its correct processes bv-deliver v first, i.e., $\Box\ (\kappa[C_{1-v}] = 0\ \wedge\ \kappa[CB_{1-v}] = 0)$.

Second, we consider an infinite sequence of bv-broadcast executions, tagged with $r \in \mathbb{N}$. Since we replace partial synchrony by our fairness assumption, the setting is fair but asynchronous, that is, processes invoke bv-broadcast infinitely many times, but at their own relative speed. Thus, they do not all invoke the bv-broadcast tagged with the same number r at the same time. Nonetheless, every process invokes bv-broadcast infinitely many times and in the r^{th} invocation its behavior depends on the messages sent in the r^{th} invocation of other processes. Therefore, we refer to the r^{th} execution of bv-broadcast even though the processes invoke it at different times.

Definition 2.2 (fairness). An infinite sequence of bv-broadcast executions is *fair* if there exists an r such that the r^{th} execution is $(r \bmod 2)$-good.

For simplicity, we use the terminology *fair* bv-broadcast when the infinite sequence of bv-broadcast executions is fair.

Algorithm 2 The Byzantine consensus algorithm at process p_i

```
1:  Global scope variable:
2:    contestants ⊆ {0, 1}, set of binary values, initially ∅.

3:  propose(est):
4:    r ← 0
5:    repeat:
6:      bv-broadcast(EST, ⟨est, i⟩)
7:      wait until (contestants ≠ ∅)
8:      broadcast(AUX, ⟨contestants, i⟩) → favorites
9:      wait until ∃c_1, ..., c_{n−t} : ∀1 ≤ j ≤ n − t favorites[c_j] ≠ ∅
10:           ∧ (qualifiers ← ∪_{∀1≤j≤n−t} favorites[c_j]) ⊆ contestants
11:     if qualifiers = {v} then
12:       est ← v
13:       if v = (r mod 2) then decide(v)
14:     else est ← (r mod 2)
15:     r ← r + 1
```

3 CONSENSUS VERIFICATION

In this section, we leverage our verification of bv-broadcast to simplify the verification of the TA of DBFT.

The Byzantine consensus algorithm. Since we replace the partial synchrony assumption by fairness, DBFT can be simplified into Algorithm 2, without coordinators and timeouts. The DBFT binary consensus invokes bv-broadcast$(\cdot)$ at line 6 and uses a set *contestants* of binary values, whose scope is global, updated by the bv-broadcast (Alg. 1, line 7) and accessed by the procedure propose$(\cdot)$ (Alg. 2, line 7).

Modeling deterministic consensus. Figure 2 depicts the TA obtained by modeling Algorithm 2 with the aforementioned method.

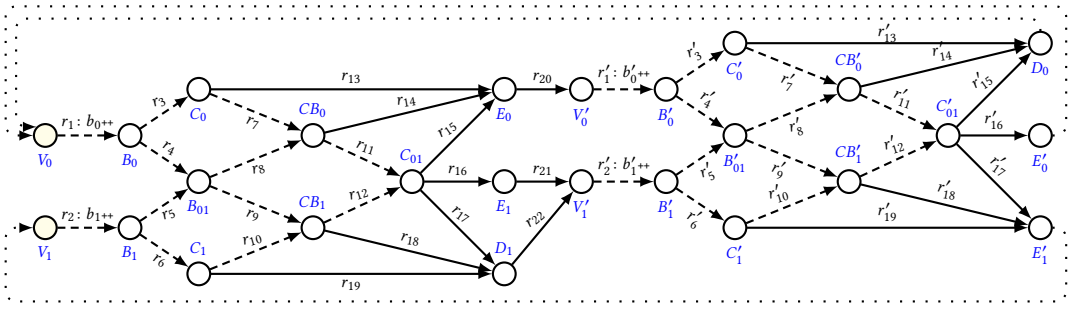

Figure 2: The naive threshold automaton of the Byzantine consensus of Algorithm 2. The embedded bv-broadcast automaton is depicted with dashed arrows. Note that the rules r_{20}, r_{21} and r_{22} represent transitions from the end of an odd round to the beginning of the following (even) round of Algorithm 2, while the dotted edges represent transitions from the end of an even round to the beginning of the following (odd) one.

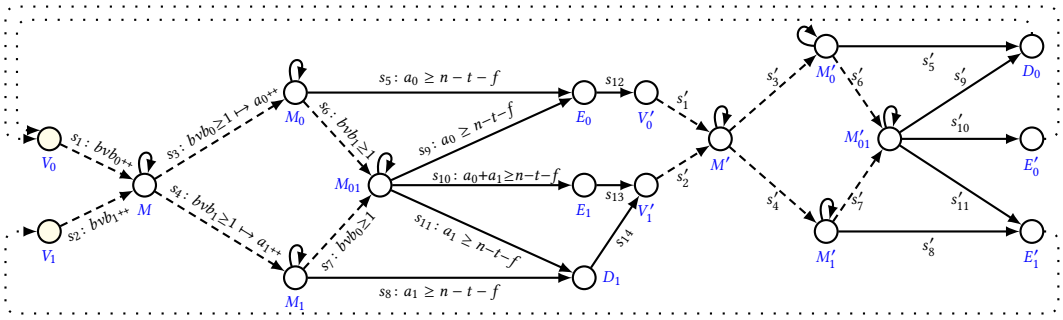

Figure 3: The simplified threshold automaton of the Byzantine consensus of Algorithm 2 obtained after model checking the bv-broadcast. Rules s'_j, $1 \le j \le 11$, are obtained from s_j by replacing each variable $c \in \{a_0, a_1, bvb_0, bvb_1\}$ with its corresponding one c'.

The TA depicts two iterations of the repeat loop (line 5), since Algorithm 2 favors different values depending on the parity of the round number. For simplicity, we refer to the concatenation of two consecutive rounds of the algorithm as a *superround* of the TA. As one can expect, this TA embeds the TA of the bv-broadcast that is depicted by the dashed arrows, just as Algorithm 2 invokes the bv-broadcast algorithm of Fig. 1. We thus distinguish the outer TA modeling the consensus algorithm from the inner TA modeling the bv-broadcast algorithm. Although Algorithm 2 is relatively simple, the global TA happens to be too large to be verified through model checking; the main limiting factor is its 14 unique guards that constrain the variables to enable rules in the TA.

Simplified threshold automaton. Our objective is to formally prove that Algorithm 2 is unconditionally safe, and that it is live under the assumption of fairness at the bv-broadcast level. Since the TA of Fig.2 is too large to be handled automatically, we build on the properties proved for the bv-broadcast to simplify in the TA from Fig.2 the part representing the bv-broadcast. We obtain Fig.3.

Verification of Byzantine Consensus. We can formally verify that DBFT (Alg. 2) solves the Byzantine consensus with the fair bv-broadcast and without partial synchrony. The full proofs can be found in the companion technical report [1]. First, we verify safety (validity and agreement) by model-checking two well-chosen superround invariants. For both values of v we obtain:

$$\forall R \in \mathbb{N}, \forall R' \in \mathbb{N}\ \Big(\Diamond \kappa[D_v, R] \neq 0 \implies \Box \kappa[D_{1-v}, R'] = 0\Big) \qquad (Agree_v)$$

$$\forall R \in \mathbb{N}\ \Big(\kappa[V_v, 1] = 0 \implies \Box \kappa[D_v, R] = 0\Big) \qquad (Valid_v)$$

Second, we verify liveness (termination) by model-checking four properties with ByMC and conclude with a short proof that relies on our fairness assumption (cf. Def. 2.2):

Theorem 3.1. *Assuming fairness of the* bv-broadcast*, Algorithm 2 terminates.*

Experiments. Thanks to our modular approach, we are able to verify all properties of the Byzantine consensus automatically in about 70 seconds whereas a non-compositional approach timed out after more than 24 hours.

REFERENCES

[1] Nathalie Bertrand, Vincent Gramoli, Igor Konnov, Marijana Lazić, Pierre Tholoniat, and Josef Widder. 2022. *Holistic Verification of Blockchain Consensus.* Technical Report 4348174. arXiv.

[2] Tyler Crain, Vincent Gramoli, Mikel Larrea, and Michel Raynal. 2018. DBFT: Efficient Leaderless Byzantine Consensus and its Applications to Blockchains. In *IEEE NCA*. 1–8.

[3] Tyler Crain, Christopher Natoli, and Vincent Gramoli. 2021. Red Belly: A Secure, Fair and Scalable Open Blockchain. In *IEEE S&P*. 466–483.

[4] Igor Konnov, Marijana Lazic, Helmut Veith, and Josef Widder. 2017. A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In *ACM POPL*. 719–734.

[5] Achour Mostéfaoui, Hamouna Moumen, and Michel Raynal. 2014. Signature-free Asynchronous Byzantine Consensus with $T < N/3$ and $O(N^2)$ Messages. In *ACM PODC*. 2–9.