Modeling and Verification of CKB Consensus Protocol in UPPAAL

Yi-Chun Feng, Yuteng Lu, Meng Sun
School of Mathematical Sciences, Peking University, Beijing, China
{yichunfeng,luyuteng,sunm}@pku.edu.cn

***Abstract*—The Nervos CKB (Common Knowledge Base) is a public permissionless blockchain designed for a peer-to-peer crypto-economy network. The CKB Consensus Protocol is a key part of the Nervos CKB blockchain that improves the Consensus's performance limit of Bitcoin. In this paper, we develop a formal model of the CKB Consensus Protocol and verify some important properties of the protocol using the UPPAAL model checker. Based on the formal model, the reliability of CKB Consensus Protocol can be guaranteed.**

***Index Terms*—CKB, Consensus Protocol, Model Checking, UPPAAL.**

I. Introduction

Blockchain can be viewed as a decentralized ledger that allows direct peer-to-peer information transfer, and has numerous benefits such as persistency, decentralization and anonymity. It has been popularized since the introduction of Bitcoin by Satoshi Nakamoto [8]. Developing a trustworthy blockchain is a very challenging task due to the complexity of the distributed execution environment and the existence of vulnerabilities. In fact, a lot of attacks on blockchains succeeded in the past years, such as the famous DAO (Decentralized Autonomous Organization) attack which results in the lost of more than 50M USD [2]. Thus, many researchers conduct investigations on blockchains' security problems.

CKB blockchain [1] has a decentralized secure layer that provides common knowledge for the peer-to-peer network. CKB Consensus Protocol [11], a key part of the CKB blockchain, is designed to overcome two Bitcoin Consensus's drawbacks: the *low transaction processing throughput* and the *vulnerability to selfish mining attacks*. CKB Consensus Protocol limits the time of connecting the sender in search of a lost transaction. This restriction can improve transaction processing efficiency without compromising the security of the blockchain. In addition, CKB Consensus Protocol adopts the "two-step confirmation", which is used to prevent selfish mining attacks.

In recent years, CKB has become very popular, and has been successfully applied in different areas. This makes the security properties of CKB more and more important. In this paper, we make some attempts to provide a formal framework for modeling CKB Consensus Protocol using timed automata and verifying its security properties in the UPPAAL model checker [4].

Formal verification has high assurance and coverage, so we choose it to analyze CKB Consensus Protocol's soundness and reliability properties. In [6], UPPAAL has been used in verification of the Bitcoin Protocol, and the probability of success of double-spending attacks based on the formal model is studied. A framework for modeling the Bitcoin contracts in Promela and use SPIN to verify whether the logic of a contract is correct is provided in [3]. The interface automata model of computation is used in [7] as a semantic domain to formalize smart contracts for detecting violations of the contract agreements. The synchronization protocol is another sub-problem for the verification of the CKB blockchain. It has been discussed in [5], [10]. Based on the related works, we could see practical meaning of combining formal verification techniques and blockchain.

The main contributions of this paper are as follows:

- We propose a formal model of the CKB Consensus Protocol in which the two-step confirmation, as well as the miner, the full node, and the block propagation, are modeled using timed automata.
- We define a family of the CKB Consensus Protocol's properties and formally verify them in the UPPAAL model checker.

The rest of this paper is organized as follows: The Nervos CKB and the CKB consensus protocol are briefly described in Section II. The formal model of the CKB consensus protocol is provided in Section III. Section IV proposes the verification of properties in UPPAAL. Section V concludes the paper and discusses possible future work.

II. A Primer on CKB

Nervos Network [1] is proposed to improve scalability and provide a better user experience. It uses the idea of off-chain to create a two-layer environment. The first layer in Nervos Network is the CKB layer. It is responsible for providing decentralized and secure infrastructure. The second layer is the environment for generating states and protecting privacy. The encryption of the first layer will protect the activities in the second layer. Under the security provided by the CKB layer, the second layer's operation can be expanded to a large extent. The operation of the Nervos Network consists of three parts: state generation executed, state verification, and storing states in the cell. After the second layer generates a new state, the state will be placed into the transaction. Later, the transactions will be broadcasted to the whole network.

The block structures in CKB include the proposal zone and the commitment zone [9], [11]. Miners put new transactions

DOI reference number: 10.18293/SEKE2021-072.

into the proposal zone after these transactions are generated. When the transactions are put into the proposal zone, the proposal step of the two-step confirmation starts. After the proposal step, the miner will put the transaction into the commitment zone and the commitment step begins. After the two-step confirmation is completed, this transaction is considered to be "valid". CKB protocol specifies that all blocks including orphan blocks that pass PoW will be broadcasted. The block propagation mechanism in the consensus protocol checks whether the transaction in the block is lost while avoiding extra round trips.

III. The Model of the CKB Consensus Protocol

Our formal model of the CKB Consensus Protocol is mainly composed of four automata: *Two-Step automaton*, *Miner automaton*, *Full Node automaton*, and *Block-Propagation automaton*. The verification is modeled based on a single transaction subject in which not only the transaction process but also the interaction among different nodes are presented.

In this model, all variables are used to mark whether the corresponding operations are successful. The default values of all variables are 0 initially. After the operations are completed, values will be assigned according to the results. The assignment is always 1 if the operation is successful. When the operation is abnormal, the variable assignment will be greater than 1. The assigned variables are treated as parameters in the constraint conditions of state transitions.

A. Two-step Automaton

The two-step confirmation consists of the "proposal" and "commitment" step. Every transaction that passes two-step confirmation will be regarded as legal. In Fig. 1, $T0$ is the initial state, representing the generation of a new transaction. The mining operation is simulated by channel *collectP!* which is synchronous with channel *collectP?* in Miner automaton. Variable *cp* is used to mark whether the transaction has been written into the proposal zone. Variable *c* is the global time, which represents the time interval of each mining epoch.

If a new transaction is processed by a miner, the variable *cp* will be assigned in Miner automaton. $T1$ represents "start of proposal confirmation". The proposal confirmation includes four operations: confirming the transaction exists in the proposal zone, checking the *txid* of the transaction, confirming the full node has received the transaction, and verifying the content of the transaction. The variable *checkT* is used to mark whether this transaction passes the *txid* check.

Channel *checkTxid!* is synchronized to *checkTxid?* in Full Node automaton. Variable *x* is the height of block on blockchain. Whenever a new block is added into blockchain, the value of *x* will increase by 1. Variable *hp* records the height of block where this transaction is located at.

In the entire two-step confirmation process, if any verification fails, the state will transfer to $T9$, indicating that the transaction cannot be broadcasted. State $T2$ means "verification of transaction". Variables *checkR* and *checkV* indicate whether the full node successfully receives and verifies the

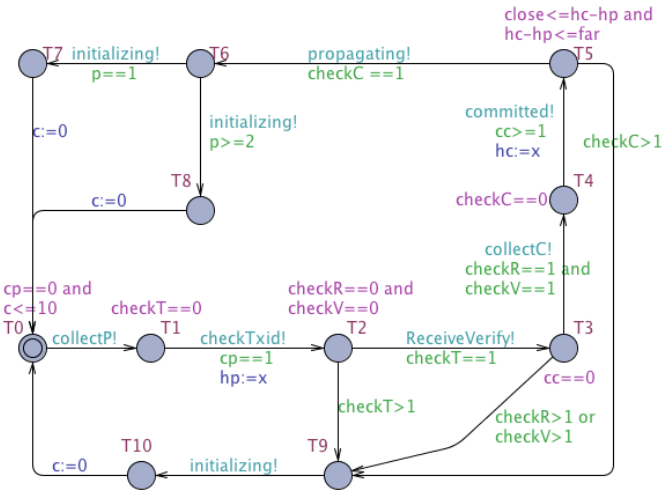

Fig. 1. The Two-step Automaton.

transaction respectively. Full Node automaton will assign values to *checkR* and *checkV* after verification. State $T3$ means that the transaction is ready for "mining of the second step". Variable *cc* is used to mark whether the transaction is written into the commitment zone. If the transaction has been verified in the first step of confirmation, it will be regarded as a "Proposed Transaction". At the same time, the transaction will be marked as proposed at height *hp* if the transaction exists in one block's proposal zone with height *hp*.

All transactions completed the first step of confirmation can be collected by miners and written into the commitment zone of a new block. Channel *collectC!* is synchronized to *collectC?* in Miner automaton. The difference in mining between the first and the second steps lies in the location where the transaction is written. The transaction will exist in two blocks with a certain height interval. The height interval must be limited within a defined range, and the interval will be checked in the second step of confirmation.

After the transaction is marked as proposed and written into the commitment zone, it will reach the state $T4$, which is "start of the commitment confirmation". This step includes two operations: confirming the proposed transaction and checking the height interval. Variable *checkC* presents whether this transaction conflicts with others on the chain. Channel *committed?* in Full Node automaton will be synchronized. When the proposed transaction enters this confirmation, it must meet the transition constraint $cc >= 1$, which means that the transaction has been written in the commitment zone. The variable *hc* marks the current height of this block on the chain.

When the state transfers to $T5$, it must conform to its invariant $close <= hc\text{-}hp <= far$. The setting of *close* is to ensure the time interval is long enough for the transaction to be propagated to the entire network. The value of *far* is designed according to the number of proposed transactions that its device can store. If the constraint condition $checkC == 1$ is met, the state can transfer to $T6$, and the channel *propagating!* will be triggered at the same time. All transactions are regarded

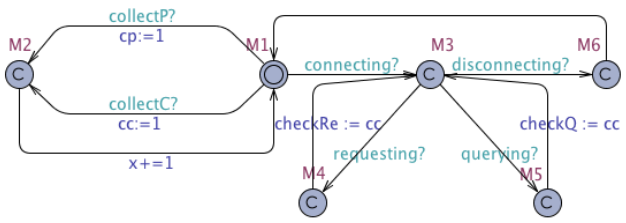

Fig. 2. The Miner Automaton.

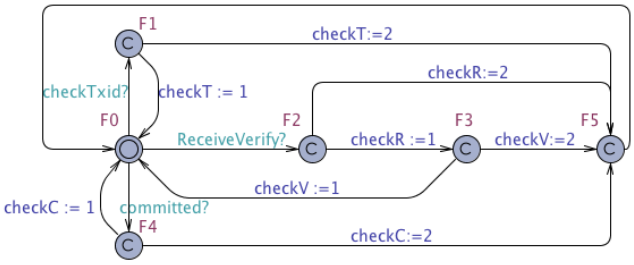

Fig. 3. The FullNode Automaton.

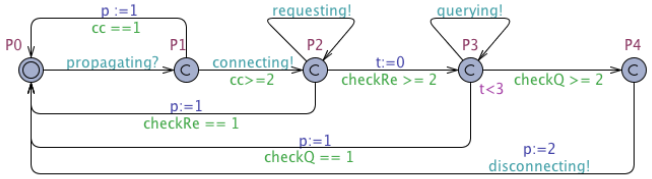

Fig. 4. The Block-Propagation Automaton.

as "Committed Transaction" when they reach state $T6$.

Channel *propagate!* is synchronized to *propagate?* in Block-Propagation automaton. If the transaction exists in the commitment zone of a certain block and is marked as proposed and committed, then the transaction can be spread to the network.

State $T7$ and $T8$ represent "authorization of broadcast" and "prohibition of broadcast" respectively. Variable p stands for whether this transaction can be propagated. Value 1 indicates that the transaction is legal and propagable, and value 2 indicates that the propagation will be blocked. The assignment of variable p will be completed by Block-Propagation automaton.

B. Miner Automaton

Fig. 2 demonstrates the behavior of a miner. $M1$ is the standby state of the miner. After mining, the state transfers to $M2$, which represents "new block generation". If there is a transaction missing during block propagation, the automaton will go to channel *connecting?* through synchronization by Block-Propagation automaton. Channel *request?* and *querying?* describe the process of the miner being asked for the missing transaction. The miner will send the requested transaction back. Variable *checkRe* and *checkQ* represent the miner's reply. Variable cc is the operation result after putting the transaction into the commitment zone. After two unsuccessful requests for the transaction, the miner will be regarded as a suspicious one and be blacklisted. This operation is simulated by the synchronization channel *disconnecting?*. State $M6$ indicates "disconnection".

C. Full Node Automaton

After the new block is generated, the full node will check the legitimacy and the PoW of blocks before broadcasting them. Fig. 3 depicts the Full Node automaton in which all operations are aimed at a single transaction. In the first step of confirmation, the full node will perform the checking of transaction *txid* and the verification of contents, which are described by the channels *checkTxid?* and *ReceiveVerify?* respectively. In the second step of confirmation, the full node is responsible for committing the transaction. $F4$ is the state after the two-step confirmation automaton synchronizes the *committed!* channel. When any operation fails, the state transfers to $F5$. At this time, the transaction is deemed invalid.

D. Block Propagation Mechanism

Fig. 4 describes the simulation of block propagation mechanism. Starting from the standby state $P0$, the entire process is started via the synchronous channel *propagating!* in Two-Step automaton. The meaning of state $P1$ is to check whether the transaction exists in the commitment zone. Variable p indicates whether this transaction can be propagated. Value 1 means that the transaction can be broadcast, and 2 means broadcast is prohibited. When cc's value is not equal to 1, it means that the transaction does not appear in any public blocks' commitment zone.

State $P2$ represents "This transaction is previously unknown." If the transaction is not found in the commitment zone of any public blocks, the synchronization channel *connecting!* must be activated to contact the miner. When *checkRe* $>= 2$, it means that the transaction is still not obtained. Then, state transfers to $P3$, which means "failure in request". Block-Propagation automaton will trigger the channel *querying!* to synchronize to the miner, and the miner must reply within a short time t. Finally, if the missing part is still not known, the automaton transfers to state $P4$. State $P4$ represents "transaction invalidation". When returning to the standby state, channel *disconnecting!* simulates the operation of blacklisting this miner and disconnecting him. At the same time, variable p is assigned to 2, which tells Two-Step automaton that this transaction should not be propagated.

IV. Verification in UPPAAL

In this section, we provide a family of properties that should be satisfied in two-step confirmation. All the properties have been verified in the UPPAAL model checker.

Property 1. All new transactions will go through the process of being placed in the proposal zone.

$$A <> TwoStep.T1$$

$T1$ means the transaction appears in the proposal zone. All newly generated transactions will be put into the proposal zone, then other nodes will receive the information about these transactions. The validity of these transactions will not affect the blocks' legitimacy and the propagation's legality.

Property 2. When a transaction is regarded as proposed, it must pass the *txid* check. If the transaction does not finish the *txid* check, it cannot become a "proposed transaction".

$$A\ [\ ]\ TwoStep.T4\ imply\ (checkT == 1)$$

$$A\ [\ ]\ not\ checkT == 1\ imply\ not\ TwoStep.T4$$

Property 3. If a transaction is proposed, the full nodes must have received and verified this transaction. If the full nodes have not received this transaction or verified the contents, this transaction cannot be regarded as proposed.

$$A\ [\ ]\ TwoStep.T4\ imply\ (checkR == 1\ and\ checkV == 1)$$

$$A\ [\ ]\ (not\ checkR == 1)\ or\ (not\ checkV == 1)\ imply\ not\ TwoStep.T4$$

In the proposal step, the first task is processing the transaction *txid*, and the second task is sending a notification to the full nodes. Before a transaction is regarded as proposed, it must pass the process of checking *txid* (*checkT* $== 1$). If the checking is unsuccessful, it will never be considered proposed. In addition, the full nodes must have received (*checkR* $== 1$) and verified (*checkV* $== 1$) this transaction. $T4$ indicates that the transaction is proposed.

Property 4. When a transaction is placed in the commitment zone, it must be received and verified by the full nodes. If the full nodes have not received and verified this transaction, it will not appear in the commitment zone.

$$A\ [\ ]\ TwoStep.T5\ imply\ checkR == 1\ and\ checkV == 1$$

$$A\ [\ ]\ not\ (checkR == 1\ and\ checkV == 1)\ imply\ not\ TwoStep.T5$$

$T5$ means the transaction is placed in the commitment zone. To activate the second stage of two-step confirmation, the transaction must be put into the commitment zone by miners. Before being put in the commitment zone, the transaction must pass the verification of the proposal step.

Property 5. When a transaction is committed, it must appear in the commitment zone with height *hc*, and satisfies the boundary: $close <= hc - hp <= far$.

$$A\ [\ ]\ TwoStep.T6\ imply\ checkC == 1\ and\ (close <= hc - hp\ and\ hc - hp <= far)$$

$T6$ is a sign of transaction commitment. The value of *checkC* indicates whether this transaction is in the commitment zone.

Property 6. If the transaction is missing, and the miner cannot obtain the transaction after requesting and querying, the miner will be disconnected and blacklisted.

$$A\ [\ ]\ BlockPropagation.P3\ and\ BlockPropagation.P4\ imply\ MiningNode.M6$$

Property 7. This model is repeatable and has no deadlock.

$$A\ [\ ]\ not\ deadlock$$

V. Conclusion and Future Work

In this paper, we formally illustrate the complete process of the two-step confirmation of the CKB consensus protocol in UPPAAL. It may lead to a better understanding of how the CKB consensus protocol works. All the properties with clear definitions are verified formally in UPPAAL. These properties could serve as a reference for the CKB application scenarios. The notion of two-step confirmation has important implications for avoiding hiding information. This may suggest various applications except for blockchain. Having acknowledged the limitations of the scope, we offer the framework of verifying the CKB consensus protocol.

In the future, we will investigate the formal model further to determine whether the CKB consensus protocol could deal with attacks. Also, the potential of its increasing throughput needs further exploration. Additional research focusing on the application scenarios would also be of great value and interest. We are hopeful that more studies with detailed results could be provided later.

Acknowledgment

This work has been supported by the Guangdong Science and Technology Department (Grant no. 2018B010107004) and the National Natural Science Foundation of China under grant no. 61772038 and 61532019.

References

[1] Nervos Network homepage, June 2020.
[2] The DAO (organization), May 2020.
[3] X. Bai, Z. Cheng, Z. Duan, and K. Hu. Formal modeling and verification of smart contracts. In *Proceedings of ICSCA 2018*, pages 322–326. ACM, 2018.
[4] G. Behrmann, A. David, and K. G. Larsen. A Tutorial on Uppaal. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *LNCS*, pages 200–236. Springer, 2004.
[5] H. Bu and M. Sun. Modeling and Verification of the CKB Block Synchronization Protocol in Coq. In *Proceedings of ICFEM 2020*, pages 287–296. Springer, Dec. 2020.
[6] K. Chaudhary, A. Fehnker, J. van de Pol, and M. Stoelinga. Modeling and verification of the bitcoin protocol. In *Proceedings of MARS 2015*, EPTCS, pages 46–60. Open Publishing Association, 2015.
[7] G. Madl, L. A. D. Bathen, G. H. Flores, and D. Jadav. Formal verification of smart contracts using interface automata. In *Proceedings of Blockchain 2019*, pages 556–563. IEEE, 2019.
[8] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008.
[9] J. Xie. Nervos CKB: A Common Knowledge Base for Crypto-Economy, Jan. 2018.
[10] Q. Zhang, Y. Lu, and M. Sun. Modeling and Verification of the Nervos CKB Block Synchronization Protocol in UPPAAL. In *Proceedings of BlockSys 2020*, pages 3–17. Springer, Nov. 2020.
[11] R. Zhang. CKB Consensus Protocol, June 2019.