Z. Zhou, D. Liu, T. R. Shmeleva and D. A. Zaitsev, "Analysis of Bitcoin Fork by Colored Petri Nets," 2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Kuching, Malaysia, 2024, pp. 4291-4296, doi: 10.1109/SMC54092.2024.10831492.

Analysis of Bitcoin Fork by Colored Petri Nets

Zeyu Zhou[1], Ding Liu[2], Tatiana R. Shmeleva[3], and Dmitry A. Zaitsev[4], *Senior Member, IEEE*

***Abstract*— Bitcoin is under the threat of fork since it operates with a distributed ledger. Predicting the fork probability in advance is beneficial for taking early action to avoid malicious attacks. In this study, we compose a colored Petri net model of Bitcoin. Our model consists of a given number of nodes, and each node has five subpages representing the node structure: proof of work, broadcast blocks, verify blocks, and the process of adding blocks to blockchain, respectively. Simulation results of fork probability can be easily obtained and analyzed by observing the data in the measuring components of subpages. The results show that our model correctly simulates the fork probability: on recent Bitcoin data, compared with the results of the wide-known SimBlock simulator, a difference of some 4.3% has been obtained. Thus, taking into account vivid graphical representation, our model has certain advantages for the developing techniques of attack avoidance.**

***Index Terms*— Bitcoin, blockchain, colored Petri Net, simulation, fork**

I. INTRODUCTION

Bitcoin, introduced by Satoshi Nakamoto in 2008 [1], is one of the most popular cryptocurrencies to date. Unlike traditional forms of currency, Bitcoin is completely decentralized, removing the central authority through a consensus mechanism so that it is not under the control of any central authority. Bitcoin system uses a distributed ledger to store its transaction data, with all users having a copy of the ledger that holds all transaction information.

Since the data of the Bitcoin blockchain is open source, any user can join the Bitcoin system and become a node in the Bitcoin network at any time. Transaction information is shared between users as broadcasted to the network and stored into a new block by miners. A new block is linked behind the previous block, thus forming the blockchain. Bitcoin has been running in a generally stable state since Satoshi Nakamoto mined the first 50 bitcoins in Finland on 3 January 2009 and is used by a large number of users every day.

Multiple copies of distributed ledger, which store the blockchain, lead to risks in the Bitcoin system due to the problem of having inconsistent data updates between ledgers [2]. Generally, a block of a certain height is mined by only one miner, and then all nodes keep the same block. In rare cases, two or more miners mine blocks of the same height in a short period of time, these blocks are kept by various nodes, which creates a blockchain fork. Overall, a blockchain fork means that various nodes of the system keep different blocks.

Forks seriously threaten Bitcoin security. When a fork occurs, miners mine the next block on their respective chains, splitting the computation power across the network, making the system more susceptible to double-spend attacks [3] and selfish mining attacks [3] launched by malicious nodes. After a while, only the blockchain that becomes the main chain will be saved, while others will be discarded along with the saved transaction records, which means forks waste a lot of computation power.

In this paper, we develop a colored Petri net [4] model to simulate the Bitcoin system. This model helps us observing the blockchains and forks easily. The simulation [5] process is visual that facilitates a comprehensive analysis of the blockchain properties. Our model is scalable, and can be used in the future to insert malicious miners or change of the network structure as required.

The rest of this article is organized as follows. Section 2 contains an overview of the Bitcoin system. Section 3 introduces the basics of the colored Petri nets. Section 4 describes our proposed model in detail. Section 5 discusses the simulation experiments conducted using the model. Section 6 represents a summary of this research and directions for future research.

II. OVERVIEW OF THE BITCOIN BLOCKCHAIN SYSTEM

Blockchain is a distributed storage database whose main features are decentralization, openness, independence, security, and anonymity [1]. This section describes the Bitcoin blockchain from three aspects: blockchain structure, proof of work, and blockchain fork.

A. Blockchain Structure

Each block of the Bitcoin blockchain consists of a block body and a block header. The block body holds information about transactions that have been verified during the block mining, which make up the blockchain's transaction ledger. The block header contains a Hash Value of the previous block, Timestamp, Nonce, Merkle Root, Version, and Bits [1]. The structure of the block is shown in Fig. 1.

Bitcoin blockchain is a chained sequence of blocks that are connected in order of generation time. If a user attempts to modify the transaction information of a block that has been added to the blockchain to attack the Bitcoin system, the Merkle Root of the block will be changed. Once the

[1]Z. Zhou is with the School of Mechano-Electronic Engineering, Xidian University, Xi'an, China (e-mail: zeyuzhou1999@163.com)

[2]D. Liu is with the School of Mechano-Electronic Engineering, Xidian University, Xi'an, China (e-mail: dliu@xidian.edu.cn)

[3]T.R. Shmeleva is with the Max Planck Institute for Software Systems, Kaiserslautern and Saarbruecken, Germany (e-mail: tatianar.shmeleva@gmail.com)

[4]D.A. Zaitsev is with the School of Computing, University of Derby, UK (e-mail: daze@acm.org)

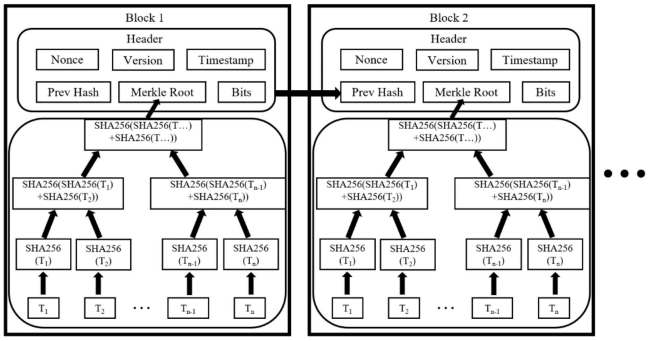

Fig. 1. Bitcoin block structure.

Merkle Root of a block is changed, the hash value of that block and all subsequent blocks will be changed, making the maliciously altered blockchain recognized by other nodes, which ensures the security of the Bitcoin system.

B. Proof of Work

The concept of proof of work was proposed by Naor and Dwork in 1993 [6] as a way to avoid spam and later used in cryptocurrencies. In the case of Bitcoin, the beginning of a block's hash consists of several consecutive zeros, forming the difficulty target of the system. A higher difficulty target means that the beginning of the block hash consists of more consecutive zeros. All the values in the block header are already set except Nonce, so miners can only change the Nonce to make the block hash value meet the difficulty target. It is impossible to derive the input value from a particular hash value due to the irreversible feature of the hashing algorithm. The only way for miners to find out the Nonce is to exhaustively enumerate it, which consumes a lot of arithmetic power. The difficulty adjustment algorithm ensures that the system produces blocks at a rate of 10 minutes [1].

C. Blockchain Fork

Bitcoin system uses a distributed ledger to store the blockchain data, which leads to the problem of inconsistent data updates between different ledgers. Here's how forks are formed:

- All miners mine at the same height simultaneously.
- Two miners mine different new blocks within a short time period and broadcast the new blocks they have mined.
- Nodes receive and save the first arriving block.
- Each node can save any of two different blocks, which means that two blockchains appear in the system.

Satoshi Nakamoto consensus mechanism states that the longest chain is the legitimate chain, but the two chains formed by the fork are both the longest, so both chains will be maintained by different miners.

The fork that becomes the longest chain ends the forked state [7], and then blocks that form a fork in another blockchain branch, become orphan blocks. As shown in Fig. 2, block B_{n+1} and block B'_{n+1} are at the same height, and

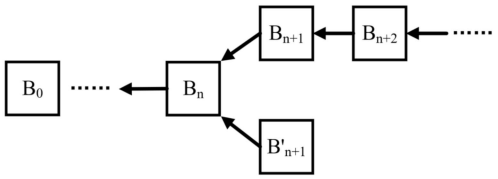

Fig. 2. Bitcoin fork scheme.

they are respectively maintained by two parts of miners. If a miner firstly proposes a new block based on block B_{n+1} with next height, block B'_{n+1} becomes an orphan block.

Miners who propose orphan blocks waste computation power and transactions recorded by orphan blocks will be revoked.

III. BASICs OF MODELING BY COLORED PETRI NETs

A place-transition net represents a bipartite directed graph with a dynamic process introduced on it. Petri [8] added dynamic elements called tokens to the model of Gill [9]. Tokens are situated in vertices of one part, called places, and moved within net as a result of transitions - the other part of vertices - firing. Recent advances in place-transition net theory, including classification of nets, are presented in [10]. Petri nets find wide application in manifold domains including automated manufacture [11], transportation, and healthcare [12]. Application of Petri nets for modeling telecommunication and networking systems are studied in [13], [14], [15].

A colored Petri net [4] represents a historical term for a special subclass of a loaded Petri net [16] where the net graph is loaded by constructs of a functional programming language ML. A token of such net represents an object of an abstract data type. Colored Petri nets of modelling system CPN Tools [17] offer the transition substitution for hierarchical design of models, and the timed delay operation to model timed characteristics. Application of formal analysis techniques for colored Petri nets, for instance, state space analysis [18], is restricted by very small models. We apply the simulation approach [5] to study colored Petri nets [4] using random distribution functions and collecting statistical data during computational experiments over rather long duration of simulated real time. For computing statistical characteristics directly in the process of simulation, a measuring components technique [4] is applied. A library of models has been developed for various networking technology that recently includes computing grids and clouds [19] and consensus protocols of cryptocurrencies [20], [21].

A colored Petri net represents a convenient tool for formal specification, modelling, verification, and simulation of complex systems, recently applied in such life-critical domain as avionics [4].

IV. DESCRIPTION OF BITCOIN MODEL

Our model is built to analyze the Bitcoin Fork Probability (BFP) using CPN Tools modeling system [17]. The overall

structure of BFP model is represented in the model main page (Fig. 3). Eleven nodes shown on the main page are represented by the same node structure subpage *node* (Fig. 4). Each *node* is structured based on the create blocks subpage (Fig. 5), broadcast blocks subpage (Fig. 6), verify blocks subpage (Fig. 7), and add blocks to blockchain subpage (Fig. 8), referred to as the *create*, *broadcast*, *verify*, and *add*, respectively. A hierarchical structure of the model is insured via the transition substitution by a subnet, the subnet name is written within a rectangle associated with the substitution transition. The node number of the model can be easily increased or decreased by copying or deleting corresponding substitution transitions.

A. Declarations of Constants, Color Sets, and Functions

Constants *n* and *m* define the total number of nodes and the total number of miners, respectively; color set *block* contains the block's height, source, the time when the block is created, and the time when it is verified; color set *bcastblock* specifies the broadcast blocks and their destinations on the Bitcoin system; color set *blockchain* represents the blockchain kept by each node; function *PoWDelay(x:real)* approximates the time taken by each miner to create a new block; function *BroadcastDelay(x:real)* approximates the network delay. The basic declarations of BFP model follows:

- *val n=11*;
- *val m=5*;
- *colset block=record h:INT* source:nnode*t0:INT* t1:INT timed*;
- *colset bcastblock=record dst: nnode*b:block timed*;
- *colset blockchain=list block*;
- *fun PoWDelay(x:real):real=real(m)* (Math.ln (1.0 -x))*(600.0)*;
- *fun BroadcastDelay(x:real):real =* (Math.ln (1.0 -x))*(12.6).*

B. Main Page of BFP Model

Fig. 3 shows the main page of the BFP model. The place *network* with the color set *bcastblock* represents network that transmits blocks between nodes. In the current model, there are 11 nodes represented by substitution transitions *node_i*. The node number and blockchain saved by each node are specified by dedicated places *nn_i* and *blockchain_i*. Each node is distinguished by its number *nn*, and after simulation, each node produces its copy of blockchain. Numbers of all the nodes are represented by the marking of a fusion set of places *nnodes*.

C. Node Structure Subpage

All the nodes of the model are represented by the same *node* subpage shown in Fig. 4. Each *node* subpage contains four subpages called *create*, *broadcast*, *verify*, and *add*. Here is the procedure of blocks' processing: a block is created by *create* subpage, and then it is broadcasted to the network by *broadcast* subpage; at the same time, a block created by a node, is verified and sent to *add* subpage to become a part of a blockchain. When the model is running, the *verify* subpage

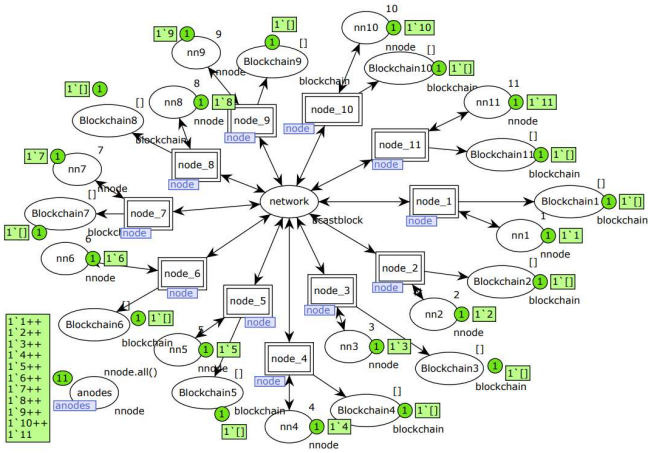

Fig. 3. Main page of BFP model.

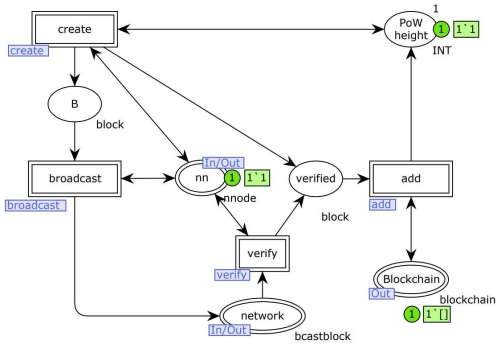

Fig. 4. Bitcoin node subpage.

constantly listens in to the network to receive the blocks transmitted to its node. Blocks received from the network are verified through the *verify* subpage and finally added to the blockchain by the *add* subpage.

It is important to note that the place *PoW height* connects to the *create* subpage and the *add* subpage. There are two ways to increase the number in place *PoW height*: the node creates a new block or receives a new block with a higher height than the number in place *PoW height*. When a node is mining a block at a certain height while receiving a verified block at the current height, the node needs to stop mining the current block and immediately start mining the block at the next height.

D. Create Block Subpage

The *create* subpage implements the process of creating blocks. In the real network, not all users are miners; place *miners* specify the numbers of miners. In Fig. 5, there are only 5 miners, but there are 11 nodes in the model. The arc inscription (1) between the transition *initial* and place *timer* means the block time.

$$clock@ + round(PoWDelay(ran())) \qquad (1)$$

Arc inscriptions (2) and (3) from transition *create* to place *verified* and place *B* represent block's details. Expression *h=Ph* represents the height of the block, *source=i* represents

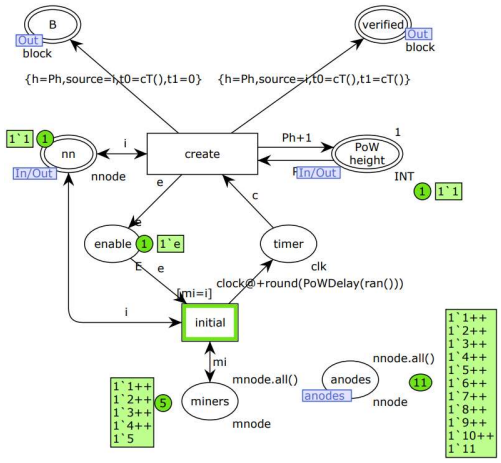

Fig. 5. Create block subpage.

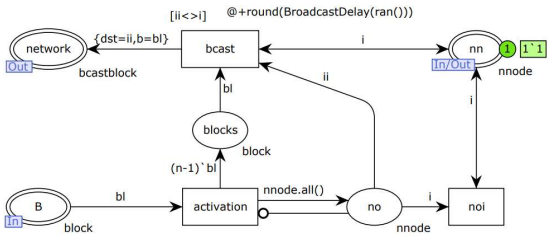

Fig. 6. Broadcast block subpage.

the node that creates the block, *t0* represents the time when the block is created, and *t1* represents the time when the block is verified. Since the block created by itself does not need to be verified, the verified time of the block is the time when it is created. The time when it is verified is unknown for the block that is going to be broadcast out, so it is set to zero.

$$h = Ph, source = i, t0 = cT(), t1 = cT() \tag{2}$$

$$h = Ph, source = i, t0 = cT(), t1 = 0 \tag{3}$$

E. Broadcast Block Subpage

New blocks created by miners are going to be broadcast to the network. Each block is bound to a destination number before broadcasting to ensure the new block is transmitted to all the other nodes. In Fig. 6, the description (4) on the transition *bcast* represents network delay, which follows an exponential distribution.

$$@ + round(BroadcastDelay(ran())) \tag{4}$$

F. Verify Block Subpage

All nodes listen to the network to receive new blocks from other nodes. The *verify* subpage in Fig. 7 represents a node that receives a new block from the network and verifies it. Description (5) in transition *check_id* checks the destination of blocks on the network. Once it finds a block's destination number matches its *nn* number, it will receive the block.

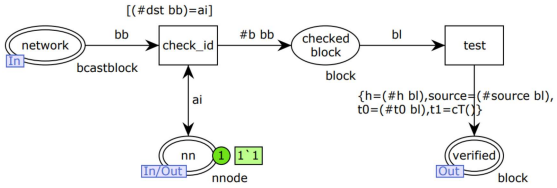

Fig. 7. Verify block subpage.

After the block is successfully verified, *t1* of the block is updated to the current time *cT()*.

$$(\#des\ bb) = ai \tag{5}$$

G. Add Block Subpage

The verified block enters the *add* subpage (Fig. 8) through transition *put*. The arc inscription of transition *put* and place *PoW height* compares the height of the verified block with the height of the place *PoW height*. When the coming block's height is not less than the height of the current proof of work on the *create* subpage, the next height of the *create* subpage is immediately updated to one greater than the current verified block, avoiding creating a block with a lower height.

The subnet consists of the transition *first coming* and the place *waiting height*, which ensures that only the first coming verified block at the same height can be saved into the blockchain. Place *waiting height* records the height of block in place *waiting*. If the block in place *incoming* with the height of one greater than the height number in place *waiting height*, it goes to the place *waiting* immediately, and the height number in the place *waiting height* is incremented. If the block in place *incoming* has the height not greater than the height number in place *waiting height*, it is an orphan block, and it will be moved to place *orphan*.

Transition *add* moves blocks from the place *waiting* into place *Blockchain* and ensures the order of blocks in the blockchain is correct through the place *block height*. Guard inscription of transition *add* ensures that blocks in the place *waiting* need to wait for six verified blocks with higher height before adding to blockchain, meaning that transactions need to wait for 6 epoch to be final.

V. ANALYSIS OF SIMULATION RESULTS

The main influences on the occurrence of forks in Bitcoin are the block time and network delay [7]. The simulation process uses these two parameters to obtain the fork probability within Bitcoin. We compose the block time random function, then verify the correctness of the model, and finally obtain the fork probability on the blockchain stretch of nearly 30000 blocks.

A. Block time function

According to the Bitcoin system's source specification, the block time fits an exponential distribution with an average of 600 seconds. The following shows the basic steps to obtain the block time function (1), i.e. an exponential distribution

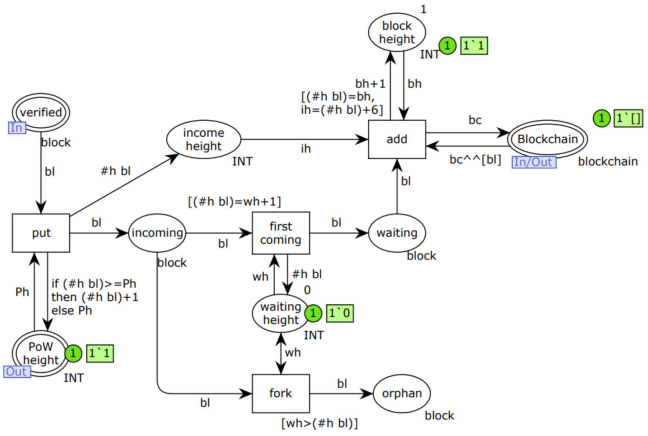

Fig. 8. Add block subpage.

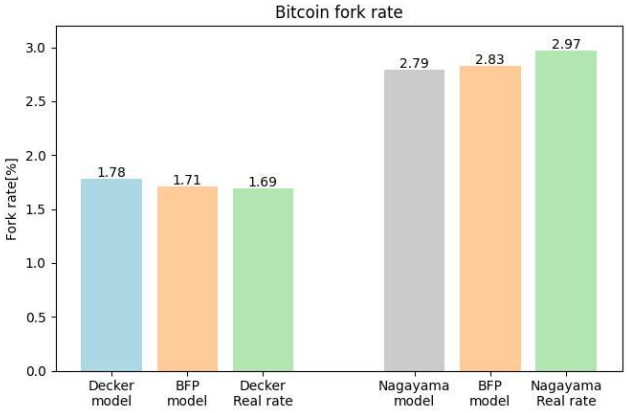

Fig. 9. Compare fork probability for: BFP model, Decker model, and Nagayama model.

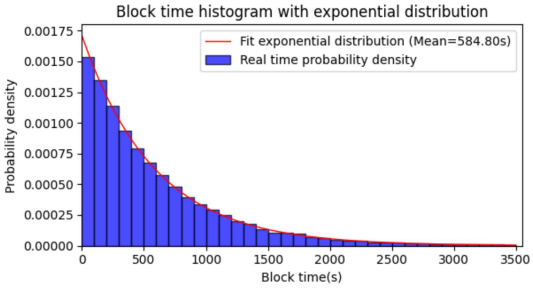

Fig. 10. Block time and fit distribution between height 804285 and 834284.

with an average of 600 seconds is (6). Once the exponential distribution is obtained, the cumulative distribution function (7) can be obtained by integrating both sides of (6).

$$f(x) = \frac{1}{600} \cdot e^{-\frac{1}{600}x} \tag{6}$$

$$F(x) = 1 - e^{-\frac{1}{600}x} \tag{7}$$

The inverse function can be calculated as (8) with **x** ranging from 0 to 1 and y ranging from 0 to positive infinity. Changing the value of x, gives us the system time to generate a new block. The number 600, representing block time, can be changed to simulate various scenarios, possibly for other cryptocurrencies as well.

$$y = -600 \cdot \ln(1 - x) \tag{8}$$

B. Fork Probability Assessment

To verify the correctness of the model, we simulated Deckers's [7] and Nagayama's [22] experiments by setting the block time function and network delay function for the model concerning their parameters. After the BFP model completes the simulation, the place *Blockchain* of each node contains the token representing the blockchain, which gives the blockchain height; the place *orphan* of the *add* subpage stores the orphan blocks generated during the operation of the system. Dividing the number of orphan blocks by the blockchain height gives us the fork probability.

To increase the confidence level of the results to 95%, 20 repetitions of the simulation were performed for each experiment, and the results are shown in Fig. 9. The fork probability difference between the BFP model and other models is within reasonable limits. Thus, the BFP model can accurately and conveniently predict forks in the Bitcoin system.

C. Evaluation of Recent Fork Probability

To obtain recent block time, we downloaded 30000 blocks between height 804285 and 834284 (from 2023/8/21 to 2024/3/11) from [23]. We got the block time distribution that is shown in Fig. 10. We found the average block time of 584.80 seconds to fit the exponential distribution as a time parameter.

Fig. 11 shows the probability density function (PDF) of block propagation delay from height 804285 to 834284 (from 2023/8/21 to 2024/3/11) [24]. We found that the real block propagation median delay is 1.098 seconds . To study the fork probability of nearly 30000 block periods, we set the block time to 584.80 seconds and the block propagation delay from 0.5s to 1.5s for the BFP model, In Fig. 12, we use a grey line to show our result as a linear relationship between network delay and fork probability which can be fitted by (9). We also show Decker's nonlinear equation [7] to compare with our results. Although the equation in known literature is nonlinear, it is approximately linear for such small network delay.

$$y = 1.4 \cdot 10^{-3} \cdot x + 4.24 \cdot 10^{-4} \tag{9}$$

In order to verify the correctness of the results obtained using the BFP model, we ran the SimBlock simulator [25] to simulate the fork probability. Network parameters were chosen from [26], [27]. Node parameters were chosen from [28] and [29]. We simulated the blockchain until the block height was 16000, and we found the fork probability was 0.188%. According to the fitted equation of (9), the fork probability of BFP model is 0.196% at 1.098 seconds, which is only 4.3% different from the above result of SimBlock simulator.

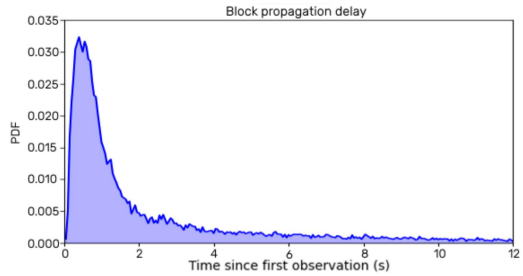

Fig. 11. Probability density function of block propagation delay.

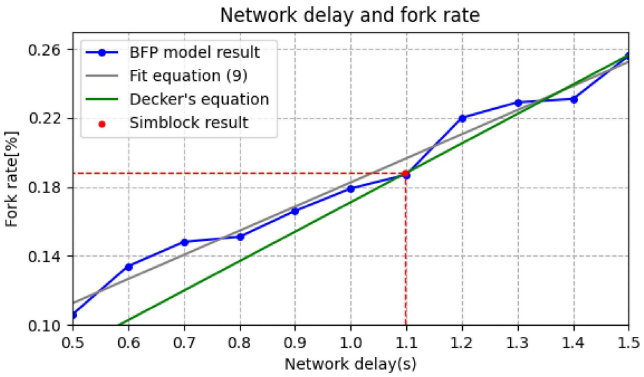

Fig. 12. Impact of network delay on fork probability.

VI. CONCLUSIONS

This paper has presented a colored Petri net model to quickly and conveniently evaluate the fork probability within Bitcoin. The subpages of the model represent the various components of the Bitcoin system, which can clearly express basic operational processes of the system, offering facilities for conveniently modifying the key parameters to simulate different operational scenarios. The key parameters of the simulation are the block time and network delay. By adjusting these two parameters and comparing the simulation results with the fork probabilities in known literature, the model composed in this study is statistically proved to be adequate.

As a future research direction, some individual miners will be transformed into malicious miners to explore the impact of malicious miners on Bitcoin security and build a reenterable model to specify a given number of nodes by a single constant. A network structure that is consistent with the topology of the real network can be added into the model, and different block propagation delay functions can be set according to the different distance between nodes.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grants Nos. 62076189, 62102285, and 62302364, the Complex Systems International Joint Research Center of Shaanxi Province, and Xi'an Theory and Applications of Discrete Event Dynamic Systems International Science and Technology Cooperation Center.

REFERENCES

[1] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
[2] Y. Zhou, X. Luo, and M. Zhou. Cryptocurrency transaction network embedding from static and dynamic perspectives: An overview. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1105–1121, 2023.
[3] S. N. Mighan, J. Mišić, V. B. Mišić, and X. Chang. An in-depth look at forking-based attacks in ethereum with pow consensus. *IEEE Trans. Netw. Service Manag.*, 21(1):507–516, 2024.
[4] D. A. Zaitsev and T. Shmeleva. *Modeling With Colored Petri Nets: Specification, Verification, and Performance Evaluation of Systems*, chapter 14, pages 378–404. IGI Global, 03 2019.
[5] Schriber T.J. *Simulation using GPSS*. New York: John Wiley , Sons Inc., 1974.
[6] M. Naor C. Dwork. Pricing via processing or combatting junk mail. In *Annual Int. Cryptology Conf.*, pages 139–147, 1993.
[7] R. Wattenhofer C. Decker. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proc.*, pages 1–10, 2013.
[8] C.A. Petri. *Kommunication mit Automaten*. PhD thesis, Technischen Hoschule Darmstadt, 1962.
[9] S. Gill. Parallel programming. *The Computer Journal*, 1(1):2–10, 1958.
[10] M. Zhou D.A. Zaitsev. From strong to exact petri net computers. *Int J Parallel Emergent Distrib Syst*, 37(2):167–186, 2022.
[11] R. Wiśniewski, J. Patalas-Maliszewska, M. Wojnakowski, and M. Topczak. Modeling and analysis of a petri net-based system supporting implementation of additive manufacturing technologies. *IEEE Trans. Autom. Sci. Eng.*, pages 1–11, 2023.
[12] J. Wang. Patient flow modeling and optimal staffing for emergency departments: A petri net approach. *IEEE Trans. Comput. Soc. Syst.*, 10(4):2022–2032, 2023.
[13] T.R. Shmeleva D.A. Zaitsev. Switched ethernet response time evaluation via colored petri net model. In *Int. Middle Eastern Multiconf. on Simulation and Modelling*, pages 68–77, 2006.
[14] A.L. Sakun D.A. Zaitsev. An evaluation of mpls efficacy using colored petri net models. In *Int. Middle Eastern Multiconf. on Simulation and Modelling*, pages 31–36, 2008.
[15] R. Yang, Z. Ding, C. Jiang, and M. Zhou. Modeling and analysis of three properties of mobile interactive systems based on variable petri nets. *IEEE Trans. Autom. Sci. Eng*, 20(4):2479–2491, 2023.
[16] A.A. Yurasov A.I. Sleptsov. *Computer-Aided Design of Flexible Computer-Aided Manufacturing Systems*. Tekhnika, Kyiv, 1986.
[17] Cpn tools. `https://cpntools.org/`. accessed Mar. 12, 2024.
[18] H. Dou, M. Zhou, S. Wang, and A. Albeshri. An efficient liveness analysis method for petri nets via maximally good-step graphs. *IEEE Trans. Syst., Man, Cybern. A, Syst*, 54(7):3908–3919, 2024.
[19] Tatiana R. Shmeleva Dmitry A. Zaitsev and Birgit Proll. Spatial specification of hypertorus interconnect by infinite and reenterable coloured petri nets. *Int J Parallel Emergent Distrib Syst*, 37(1):1–21, 2022.
[20] B. Pröll, W. Retschitzegger, W. Schwinger, T. R. Shmeleva, and D. A. Zaitsev. Modelling proof-of-work agreement protocol by coloured petri nets. *Int J Parallel Emergent Distrib Syst*, 37(6):597–612, 2022.
[21] D.A. Zaitsev, T.R. Shmeleva, Z. Zhou, and D. Liu. Verification of cryptocurrency consensus protocols: reenterable colored petri net model design. *Int J Parallel Emergent Distrib Syst*, 39(1):32–50, 2024.
[22] K. Shudo R. Nagayama, R. Banno. Identifying impacts of protocol and internet development on the bitcoin network. In *IEEE Symp. on Computers and Communications (ISCC)*, pages 1–6, 2020.
[23] Blockchair database dumps. `https://gz.blockchair.com/bitcoin/blocks/`. accessed Mar. 12, 2024.
[24] Bitcoin monitoring. `https://www.dsn.kastel.kit.edu/bitcoin/index.html`. accessed Mar. 26, 2024.
[25] Simblock. `https://dsg-titech.github.io/simblock/`. accessed Mar. 12, 2024.
[26] Testmy.net. `https://testmy.net/country`. accessed Mar. 12, 2024.
[27] Global ping statistics. `https://wondernetwork.com/pings`. accessed Mar. 12, 2024.
[28] Network snapshot. `https://bitnodes.io/nodes/`. accessed Mar. 12, 2024.
[29] Blockchain charts. `https://www.blockchain.com/explorer/charts`. accessed Mar. 26, 2024.