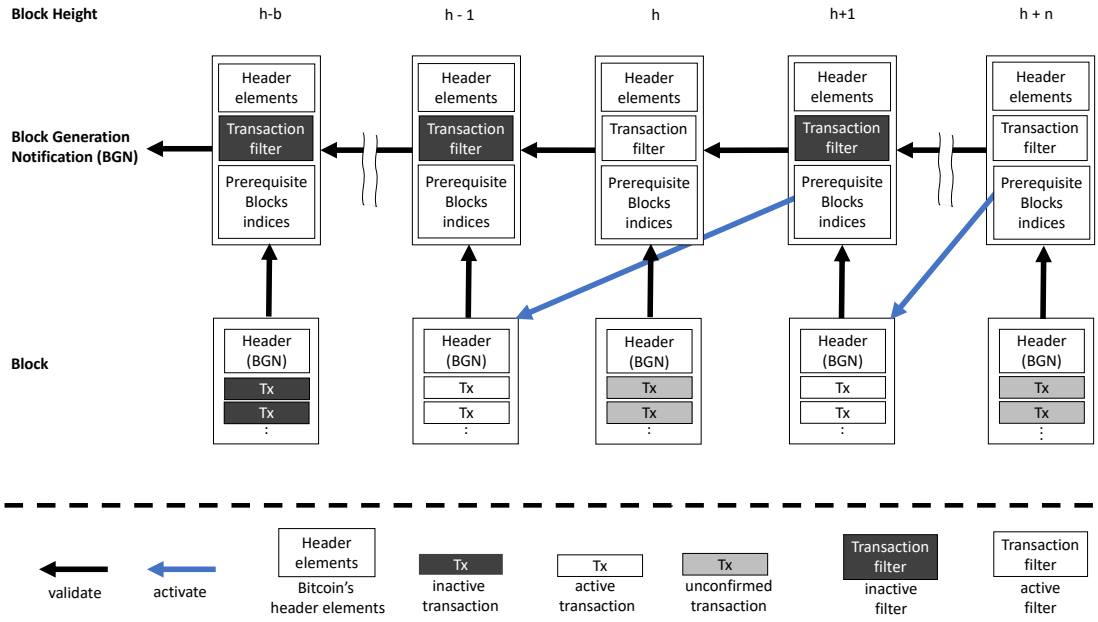

Fig. 1. Summary of the proposed protocol.

Algorithm 2 Validation of block B_h

1: Block generation notification N_h id validated.
2: All transactions in block B_h are checked to ensure that they pass through the integration filter I_h.
3: Each transaction in block B_h id validated.

Algorithm 3 Creation of integration filter I_h

1: An integration filter I_h = an empty Bloom filter that is a bit array, all set to 0.
2: $r = h$
3: **while** $r \geq 0$ **do**
4: **if** F_r is not inactivated **then**
5: $T_h = Unite(T_h, F_r)$ ▷ $Unite$ is a function that takes the logical OR of each bit.
6: **end if**
7: $r = r - 1$
8: **end while**
9: **return** T_h

to ensure that it has been validated. Thus, to validate N_h, it is recursively required that all previous block generation notifications have been validated. Second, the prerequisite block indices in N_h are checked, and the activated blocks are validated. If the node receiving N_h has not received these blocks, the node requests them for the N_h sender. Last, each header element of N_h are validated. In this validation, the Merkle root is not validated because block B_h is required for this validation; otherwise, the same is true for the validation of the Bitcoin header. Block B_h is not validated in the N_h validation.

D. Validation of Block

Algorithm 2 shows the algorithm for validation of block B_h of height h. The node receiving a block generation notification higher than height h activating block B_h validates block B_h.

In the validation of block B_h, first, the corresponding block generation notification N_h is validated. Second, the next step is to check that all transactions in block B_h pass through the integration filter I_h. The integration filter is the integration of the active transaction filters at height h, and the algorithm for creating the integrated filter is given by Algorithm 3. Last, each transaction in block B_h is validated. The validation of one previous block B_{h-1} is not performed because in the proposed protocol, blocks are activated by a block generation notification, and the prerequisite block when block B_h was generated is indicated in block generation notification N_h and is validated in the first N_h verification (Algorithm 1).

E. Mining

Algorithm 4 shows the algorithm by which the node receives block generation notification N_h and starts the next mining.

When ntheode that received a block generation notification N_h starts the next mining, it validates the received N_h (Algorithm 1). Then, if any block B_i up to the previous b blocks already received has not yet been activated, the corresponding transaction filter F_i is inactivated, and the prerequisite block indices are updated (there can be multiple blocks to be activated). Subsequently, an integrated filter I_h is generated (Algorithm 3). The generated integrated filter I_h is then used to determine transactions in the transaction pool of the receiver. The UTXOs consumed by each transaction are determined by the integrated filter I_h and at least one positive transaction is excluded from mining. From the remaining transactions,

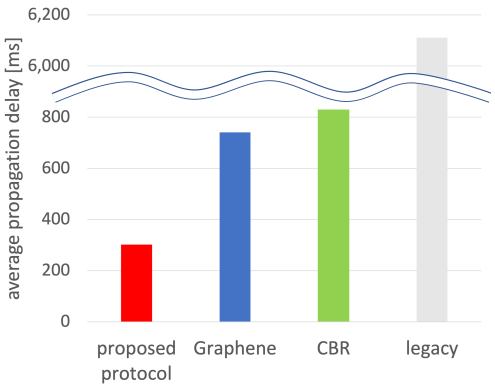

Fig. 3. Propagation time for each percentile.

TABLE VIII
PROPAGATION TIME.

	Proposed protocol	Graphene	CBR	legacy
50 %ile propagation time [ms]	274	666	746	6182
90 %ile propagation time [ms]	453	1154	1241	8633
100 %ile propagation time [ms]	634	2282	2367	15326
Average propagation time [ms]	302	741	828	6478

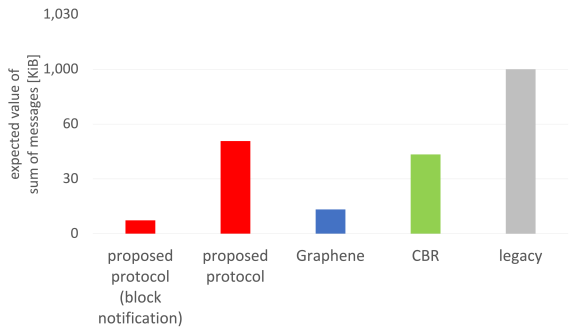

Fig. 4. Expected message size between two nodes per hop.

Figure 5 shows the fork rate. We measure the propagation time of the block generation notification for the proposed protocol. These results show that the proposed method had the best fork rate, which is 40.8 % of that of Graphene.

V. CONCLUSION

In this paper, we proposed a protocol for quickly propagating block generation notification using a Bloom filter in the blockchain network to reduce the fork rate. In experiments in which a simulator is used, the propagation time of the 50 %ile was 41.1 % of an existing protocol, that of the 90 %ile was 39.2 % of the existing protocol, and the fork rate calculated from the average propagation time is 40.8 % of the existing protocol. Although shortening the block generation interval increases the fork rate [2], it can be offset by suppressing the fork rate using the proposed protocol, so that the block generation interval can be shortened while maintaining the fork rate. Arakawa et al. [14] proposed such a method.

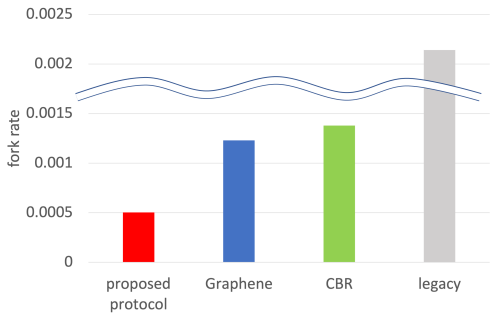

Fig. 5. Fork rate.

TABLE IX
FORK RATE

	Proposed protocol	Graphene	CBR	legacy
T_W [ms]	302	741	828	6478
F	0.000503	0.00123	0.00138	0.0107

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number JP21H04872.

REFERENCES

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, page 21260, 2008.
[2] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*, pages 507–527. Springer, 2015.
[3] Matt Corallo. Bip 152: Compact block relay, 2016. https://github.com/Bitcoin/bips/blob/master/bip-0152.mediawiki.
[4] A Pinar Ozisik, Gavin Andresen, Brian N Levine, Darren Tapp, George Bissias, and Sunny Katkuri. Graphene: efficient interactive set reconciliation applied to blockchain propagation. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 303–317. 2019.
[5] Blockchain.com, (Accessed on 10/12/2022). https://www.blockchain.com/explorer/charts.
[6] Ryunosuke Nagayama, Ryohei Banno, and Kazuyuki Shudo. Identifying impacts of protocol and internet development on the bitcoin network. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020.
[7] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
[8] Michael T Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 792–799. IEEE, 2011.
[9] Bitcoin visuals, (Accessed on 10/12/2022). https://Bitcoinvisuals.com/chain-input-count-tx.
[10] Kaylash C Chaudhary, Vishal Chand, and Ansgar Fehnker. Double-spending analysis of bitcoin. In *Pacific Asia conference on information systems*. Association for Information Systems, 2020.
[11] Yusuke Aoki, Kai Otsuki, Takeshi Kaneko, Ryohei Banno, and Kazuyuki Shudo. Simblock: A blockchain network simulator. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 325–329. IEEE, 2019.
[12] Ryohei Banno and Kazuyuki Shudo. Simulating a blockchain network with simblock. In *2019 IEEE international conference on blockchain and cryptocurrency (ICBC)*, pages 3–4. IEEE, 2019.
[13] Akira Sakurai and Kazuyuki Shudo. Impact of the hash rate on the theoretical fork rate of blockchain. In *2023 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–4. IEEE, 2023.
[14] Masumi Arakawa and Kazuyuki Shudo. Block interval adjustment based on block propagation time in a blockchain. In *2022 IEEE International Conference on Blockchain (Blockchain)*, pages 202–207. IEEE, 2022.