Enhanced Byzantine Fault Tolerance with Raft and Multi-Pipeline HotStuff Using ECC

Shen Li[a]
School of Information Systems Engineering and Management, Harrisburg University of Science and Technology, Harrisburg, U.S.A.

Keywords: Byzantine Fault Tolerance, Raft-MPH, Elliptic Curve Cryptography, HotStuff.

Abstract: This paper focuses on the challenges in Byzantine Fault Tolerance (BFT) systems, particularly focusing on the inefficiencies in traditional protocols such as Practical Byzantine Fault Tolerance (PBFT) and the shortcomings of HotStuff's single-pipeline design. This study introduces a new model called Raft-Multiple Pipeline HotStuff (Raft-MPH) with Elliptic Curve Cryptography (ECC) to deal with BFT. Additionally, the Raft-MPH protocol is designed to improve HotStuff's existing framework by leveraging ECC to make cryptographic operations more efficient. The research shows that this approach significantly decreases communication overhead while maintaining high throughput and low latency, even in different network conditions. The Raft-MPH protocol processed up to 160K transactions per second (TPS) on a Local Area Network (LAN) with latency similar to traditional HotStuff, and it scaled well as the number of replicas increased. Overall, this work lays a solid foundation for future research on adaptive consensus protocols and may lead to practical applications in blockchain platforms.

1 INTRODUCTION

Byzantine Fault Tolerance (BFT) has captured significant attention in distributed systems, particularly as blockchain technology demands robust safety and reliability, even when dealing with malicious nodes. Facing these challenges, researchers utilize these BFT protocols to ensure smart contract execution and secure consistent state machine replication in these networks, maintaining system integrity even when malicious behaviors are detected. However, traditional BFT approaches, such as Practical Byzantine Fault Tolerance (PBFT), struggle in the large-scale distributed environment due to their costly view-change processes and high communication complexity (Bogdanov et.al, 2023). Introducing more sophisticated protocol, like HotStuff, handles these challenges by optimizing consensus phases to reduce forks and achieve linear communication complexity (Yin et.al, 2018; Niu et.al, 2021). Despite these enhancements, HotStuff's single-pipeline architecture introduces new research gaps, necessitating further exploration. This paper proposes the Raft-Multiple Pipeline HotStuff (Raft-MPH) model, which combines Raft's simplicity with HotStuff's multiple-pipeline architecture and Elliptic Curve Cryptography (ECC) to improve performance, scalability, and fault tolerance.

The field of Byzantine Fault Tolerance has achieved significant results, with diverse protocols developed to deal with the inherent issues of maintaining consensus in distributed systems. PBFT, one of the most well-known and earliest BFT protocols, has strong fault tolerance management capabilities but suffers from quadratic communication complexity O (n^2), making it less feasible in large-scale distributed systems. The cost of view changes in PBFT, which grows cubically O (n^3), further undermines its capabilities to handle scalability issues (Bogdanov et.al, 2023). In response, HotStuff has been introduced to simplify the consensus process by decreasing the number of phases required and achieving linear communication complexity in steady-state operations. HotStuff's approach, which utilizes threshold signatures, has played a crucial role in minimizing latency and improving performance (Yin et.al, 2018; Niu et.al, 2021). However, the inherent drawbacks of a single-

[a] https://orcid.org/0009-0008-2726-2728

Li and S.
Enhanced Byzantine Fault Tolerance with Raft and Multi-Pipeline HotStuff Using ECC.
DOI: 10.5220/0013524400004619
In *Proceedings of the 2nd International Conference on Data Analysis and Machine Learning (DAML 2024)*, pages 379-384
ISBN: 978-989-758-754-2

pipeline mechanism limit throughput during heavy transaction demands. MPH was developed to address these challenges by allowing concurrent proposal and voting processes, thereby significantly improving throughput without introducing delay (Cheng et.al, 2022).

Moreover, integrating Raft with ECC has been explored to improve encryption efficiency and ease operational tasks (Lahraoui et.al, 2024; Lara-Nino et.al, 2018). However, existing solutions still struggle with single-thread limitations and communication overhead, particularly in Byzantine fault-prone environments. This research builds upon these previous achievements by introducing a hybrid model that dynamically changes between Raft and MPH modes, further improving performance and resilience in distributed systems.

This research aims to develop the Raft-MPH protocol with ECC to tackle major challenges in BFT systems, including reducing communication overhead, enhancing fault tolerance, and improving scalability. It combines Raft's leader-based system with MPH's multi-threaded processing to (1) lower communication complexity and (2) integrate their advantages for high throughput and fault tolerance (3) This protocol dynamically adapts based on network settings and uses ECC for cryptographic efficiency. This research improves scalability and security in distributed systems, especially in blockchain, and paves the way for future work on practical implementation and optimization of adaptive consensus protocols.

2 METHODOLOGIES

2.1 Threat Models

The Raft-PBFT model with ECC aims to tackle issues related to Byzantine fault tolerance and improve cryptographic efficiency in distributed systems. However, this model has several vulnerabilities that could compromise its effectiveness (Figure 1): (1) Byzantine failures remain a critical concern due to Raft's leader-based election process combined with PBFT's view-change mechanism. Malicious nodes can disrupt the consensus process, which leads to increased latency and undermines the system's reliability. (2) Crash-stop failure is another issue because the model relies on a single pipeline. The entire system's performance will be severely impacted if a node fails. This will lead to system-wide delays. (3) The attackers could exploit a timing vulnerability caused by dynamic mode switching between Raft and PBFT modes. This will lead to instability and compromised consensus (Bogdanov et.al, 2023).

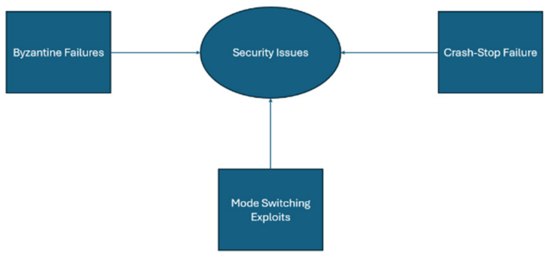

Figure 1: The pipeline of security issues (Picture credit: Original).

Based on the threat models of the current approach, this research proposes the Raft-MPH consensus protocol with ECC, which combines the principles of Raft and ECC with the Multiple pipeline approach in HotStuff to improve system performance, safety, and liveness.

2.2 Methods

2.2.1 Raft

Raft is a consensus algorithm designed for its simplicity and reliability in managing a replicated log across a cluster of nodes. It divides the consensus problem into three components: leader election (Figure 2), log replication, and safety (Ongaro and Ousterhout, 2014; Zhan and Huang, 2023). (1) Raft elects a single server as the leader to manage the log replication process. If the leader fails, the followers start an election to elect a new leader. Randomized election timeouts prevent split votes, which secure smooth leadership transitions. (2) The leader takes log entries from clients and broadcasts them to the followers. Once most followers have received an entry, it is considered committed and is applied to the state machine. (3) Raft ensures that all nodes utilize the same log entries in order, even during failure. The leader is responsible for dealing with inconsistencies by correcting the followers' logs. Thus, Raft's strong leadership, log replication, and consistent consensus model effectively handle Byzantine failures, crash-stop failures, and vulnerabilities related to dynamic mode switching.

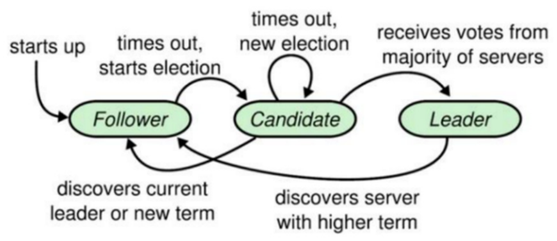
starts up
times out, starts election
times out, new election
receives votes from majority of servers
Follower
Candidate
Leader
discovers current leader or new term
discovers server with higher term

Figure 2: Raft election role replacement process (Picture credit: Original).

2.2.2 MPH

The MPH is designed to improve performance and scalability in distributed systems, especially those vulnerable to Byzantine failures. Unlike the sequential nature of HotStuff, MPH optimizes the process by enabling simultaneous proposing and voting. This optimization decreases the bottleneck related to single-pipeline methods. MPH can adapt to real-time network conditions by operating under a partially synchronous communication model. A major benefit of MPH is its ability to maintain linear communication complexity during the normal operation phase while achieving quadratic complexity during view changes due to its multi-pipeline approach. This design enables more blocks to be proposed and committed in each round, significantly enhancing throughput without increasing end-to-end latency. Moreover, MPH's optimistic responsiveness allows a correct leader to reach a consensus quickly (Cheng et.al, 2022). This will improve the protocol's resilience against Byzantine failures, reduce the impact of node failures, and simplify leader election processes to prevent vulnerabilities during mode transitions.

2.2.3 ECC

Compared to traditional algorithms like RSA, ECC has smaller key sizes. It further secures the system by offering secure encryption, Digital signatures, and key exchange. Three characteristics offer more robust security with lower computational costs: (1) ECC depends on the elliptic curve equation over a finite prime field, where security is based on the difficulty of solving the elliptic curve discrete logarithm problem (ECDLP). (2) The Elliptic Curve Diffie-Hellman (ECDH) protocol ensures secure key exchange over insecure channels. (3) ECC supports digital signatures via protocols like the Elliptic Curve Digital Signature Algorithm (ECDSA), securing message integrity and authenticity (Lahraoui et.al, 2024). Therefore, ECC plays an important role in mitigating Byzantine failures and reducing delays from crash-stop failures due to its lightweight design. It also ensures smooth transitions between Raft and PBFT modes, which keeps the system stable and secure.

2.2.4 Raft-MPH with ECC

The Raft-MPH consensus protocol with ECC operates within a partially synchronous network. The nodes use a consensus protocol in this system to agree on the transaction sequence. The system assumes up to f faulty nodes out of n, where $n > 3f + 1$ (Cheng et.al, 2022). The protocol has two phases: Normal Case Operation and View-Change Operation (Cheng et.al, 2022). An honest leader is assumed to work within a synchronous network during Normal Case Operations. The protocol uses a multi-pipeline scheme to ensure simultaneous proposals and voting. ECC improves cryptographic efficiency, making the system process more transactions smoothly. However, View-Change Operation is triggered through timeouts if the system detects a malicious leader or encounters network asynchrony. This allows the system to elect a new leader and update the new view number. Additionally, Raft-MPH supports parallel proposals and voting across different views, significantly enhancing throughput. The protocol also applies a 3-chain commit rule (Figure 3) for safety, allowing blocks to be committed only as part of a chain of three consecutive certified blocks with valid quorum certificates (Cheng et.al, 2022). Combined with the view-change process, this mechanism maintains system consistency and progress, even under hostile conditions.

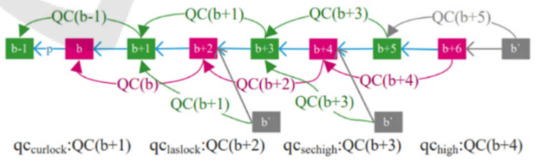

Figure 3: 3-Chain predicate of MPH (Picture credit: Original).

The correctness of Raft-MPH with ECC is proven through eight lemmas that ensure the safety and liveness of the protocol. (1) Lemma 1 claims that no two Quorum Certificates (QCs) can have the same view number, preventing conflicting votes. (2) Lemma 2 asserts that if two certified blocks share the same view number, they must be the same. This prevents conflicting commitments. (3) Lemma 3 secures that a block is committed only if it extends from a chain of three consecutive certified blocks. (4)

Lemma 4 states that only one block can be certified in view. (5) Lemma 5 indicates nodes will move to the next view upon receiving a correct leader's proposal. (6) Lemma 6 states that all correct nodes will push opinions monotonically after the Global Stabilization Time (GST). (7) Lemma 7 ensures correct leader proposals will be received and voted on, even in malicious conditions. (8) Lemma 8 guarantees a block will be certified and committed, preserving liveness and enabling transaction processing (Yin et.al, 2018; Niu et.al, 2021; Cheng et.al, 2022).

2.3 Implementation Details

In this project, Java implements the Raft-MPH protocol. It manages concurrency across multiple pipelines using Java's built-in concurrency utilities. This process enables parallel execution of proposals and votes. Java.net libraries handle networking with Netty for efficient communication. Implemented through Bouncy Castle, ECC uses the secp256k1 curve for secure key exchanges and digital signatures. This protocol also assumes up to f malicious nodes out of n total nodes ($n > 3f + 1$). Java's collections manage logs and QC, and exception handling and retry mechanisms secure resilience during Byzantine and crash-stop failures. Finally, Java's testing frameworks validate protocol safety and liveness.

3 RESULT AND DISCUSSION

In this chapter, the author evaluates the results of the Raft-MPH protocol integrated with ECC in a BFT system. This evaluation focused on three major areas: (1) throughput and latency performance, (2) scalability, and (3) fault tolerance under different network conditions. Research demonstrates how the proposed Raft-MPH with ECC protocol effectively reduces communication overhead, improves fault tolerance, and enhances scalability in distributed systems.

3.1 Throughput and Latency Analysis

As indicated in Figure 4 and Table 1, the throughput and latency of Raft-MPH with ECC were tested under two network settings: Local Area Network (LAN) and Wide Area Network (WAN). This study used a block size of 800 transactions, each with a payload of 1024 bytes. The mempool dissemination batch size was set to 512 KB to ensure efficient transmission.

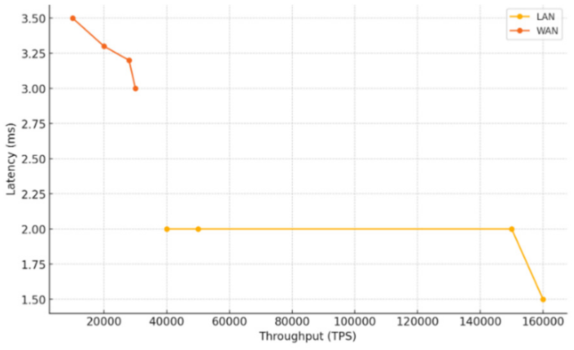

Figure 4: Throughput (TPS) vs. Latency (ms) for Raft-MPH with ECC on LAN and WAN (Picture credit: Original).

The results showed that this proposed protocol maintains low latency while achieving high throughput under both network conditions. Specifically, Raft-MPH can process up to 160k transactions per second (TPS) on a LAN while maintaining latency similar to HotStuff. However, due to restricted bandwidth and the challenges of long-distance transactions, the maximum TPS on a WAN was limited to 30k, with an increased latency. The multi-threaded processing in Raft-MPH with ECC enables efficient bandwidth utilization, resulting in a substantial enhancement in throughput compared to traditional protocols like HotStuff and PBFT. Deploying ECC for cryptographic operations helped reduce communication overhead, as ECC's smaller key sizes and faster computation times enabled more transactions to be processed within the same time slot. Thus, with the integration of Raft's leader-driven election process with MPH's multi-threaded approach, this protocol keeps throughput consistently high, even as the network scales.

Table 1: Summary of throughput and latency performance metrics.

Network Environment	Block Size	Payload Size (Bytes)	Mempool Dissemination Batch Size (KB)	Throughput (TPS)	Latency (ms)
LAN	800	1024	512	Up to 160k	Comparable to HotStuff (~2s)
WAN	800	1024	512	Up to 30k	Higher than LAN (~3.5s)

Table 2: Scalability metrics for Raft-MPH with ECC.

Number of Replicas	Maximum Throughput (TPS) on LAN	Maximum Latency (ms) on LAN	Maximum Throughput (TPS) on WAN	Maximum Latency (ms) on WAN
4	160k	~1.5	30k	~3
10	150k	~2	28k	~3.2
22	50k	~2	20k	~3.3
58	40k	~2	10k	~3.5

Table 3: Fault tolerance performance metrics for Raft-MPH with ECC.

Number of Faulty Replicas	Throughput (TPS) on LAN	Latency (ms) on LAN	Throughput (TPS) on WAN	Latency (ms) on WAN
0	50k	~1.5	20k	~3
1	48k	~1.8	18k	~3.2
2	40k	~2	15k	~3.4
3	30k	~2	10k	~3.5

3.2 Scalability

Figure 5 and Table 2 describe the scalability results of Raft-MPH with ECC when the number of replicas increases from 4 to 58. The protocol was tested under the same network environments to maximize throughput and latency at various scales.

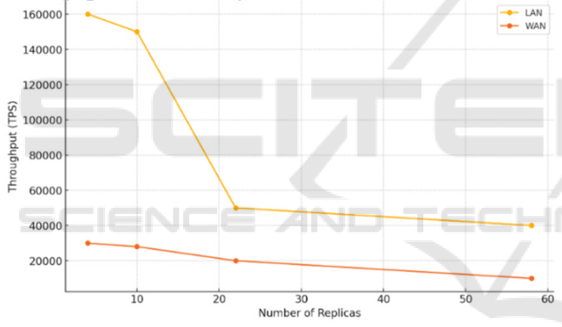

Figure 5: Throughput (TPS) vs. Number of Replicas for Raft-MPH with ECC (Picture credit: Original).

Research revealed that as more replicas were added, the throughput of Raft-MPH with ECC scaled smoothly, achieving 40k TPS on a LAN with 58 replicas while keeping the average latency at 2 milliseconds. This performance is about 60% better than HotStuff achieves at every scale. Due to Raft-MPH's linear communication cost, O(n), with ECC's efficient cryptographic operations. Research has managed to keep latency growth to a minimum as the system expands. This gives this protocol a clear advantage over protocols like PBFT, which struggle with quadratic communication costs, O(n^2).

3.3 Fault Tolerance and View-Change Performance

Figure 6 and Table 3 present the performance of Raft-MPH with ECC under fault-tolerant scenarios. This research tested the protocol with up to 3 faulty replicas in a 22 replicas setup on a LAN, where the system dealt with leader failures by performing view-change operations.

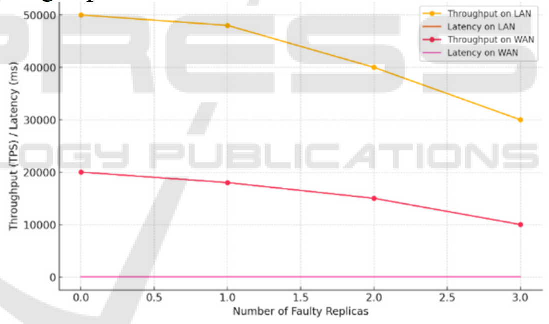

Figure 6: Throughput (TPS) and Latency (ms) vs. Number of Faulty Replicas for Raft-MPH with ECC (Picture credit: Original).

The results indicated that Raft-MPH with ECC outperformed both HotStuff and PBFT, even when the number of faulty replicas increased. Additionally, the multi-pipeline mechanism enables the protocol to maintain higher throughput and lower latency, even when handling view-change operations. ECC improves fault tolerance by securing the communication between replicas, so even if some replicas fail, the system can quickly elect a new leader and resume normal operations. Even with 3 faulty replicas, the maximum latency was measured at 2 milliseconds, and throughput remained at around 30k TPS. The combination of ECC ensures that the protocol maintains cryptographic security, preventing unauthorized access and manipulation of the system, even during view-change processes.

3.4 Takeaways

The research results reveal that the Raft-MPH protocol with ECC efficiently handles the major changes in the BFT system, including decreasing communication overhead, improving fault tolerance, and enhancing scalability. This protocol's ability to adjust to changing network conditions, along with the efficiency of ECC, makes it a reliable solution for modern distributed systems, particularly in blockchain applications. This research proves that Raft-MPH with ECC provides enhancement over traditional consensus protocols. It builds a solid foundation for further research and practical implementation in high-performance, secured distributed systems.

4 CONCLUSIONS

In this research, the author proposed Raft-MPH with ECC to handle the challenges of performance, scalability, and BPT consensus mechanisms. This approach integrates Raft's simplicity with HotStuff's multi-pipeline architecture and ECC's cryptographic efficiency. Additionally, research results demonstrated that Raft-MPH with ECC significantly reduces communication overhead, enhances throughput, and improves cryptographic operations compared to traditional BFT protocols like PBFT and HotStuff. Future work will translate Raft-MPH with ECC from theory into practice by developing a robust implementation framework and integrating it into blockchain platforms. This includes continuous improvements in adaptive consensus mechanisms, applying advanced machine learning models such as the Adaptive Weighted Attribute Propagation (AWAP) model, which can dynamically switch between Raft and MPH modes based on real-time network environments (Xue et.al, 2021). Moreover, exploring advanced cryptographic techniques will ensure the protocol's reliability and security in tandem with technological advancements (Salam et.al, 2024). These efforts will pave the way for Raft-MPH with ECC to become a practical and efficient solution for modern distributed systems.

REFERENCES

Bogdanov, A., Shchegoleva, N., Khvatov, V., et al. 2023. The Combination of P-BFT and RAFT: A New Approach to Building Networks that Provide Reliability and Security. International Conference on Computational Science and Its Applications. Cham: Springer Nature Switzerland, 2023: 572-583.

Cheng, T., Zhou, W., Yao, S., et al. 2022. Multi-pipeline HotStuff: A High Performance Consensus for Permissioned Blockchain. IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 1008-1019.

Lahraoui, Y., Lazaar, S., Amal, Y., et al. 2024. Securing Data Exchange with Elliptic Curve Cryptography: A Novel Hash-Based Method for Message Mapping and Integrity Assurance. Cryptography, 8(2), 23.

Lara-Nino, C. A., Diaz-Perez, A., & Morales-Sandoval, M. (2018). Elliptic curve lightweight cryptography: A survey. IEEE Access, 6, 72514-72550.

Niu, J., Gai, F., Jalalzai, M.M., et al. 2021. On the performance of pipelined hotstuff. Conference on Computer Communications, 1-10.

Ongaro, D., Ousterhout, J., 2014. In search of an understandable consensus algorithm. USENIX annual technical conference, 305-319.

Salam, A., Abrar, M., Amin, F., et al. 2024. Securing smart manufacturing by integrating anomaly detection with zero-knowledge proofs. IEEE Access.

Xue. S.X., Ji, M.W., Jun, Y.Y., et al. 2021. AWAP: Adaptive weighted attribute propagation enhanced community detection model for bitcoin de-anonymization. Applied Soft Computing, 109, 107507.

Yin, M., Malkhi, D., Reiter, M.K., et al. 2018. HotStuff: BFT consensus in the lens of blockchain. arXiv preprint: 1803.05069.

Zhan, Z., Huang, R., 2023. Improvement of Hierarchical Byzantine Fault Tolerance Algorithm in RAFT Consensus Algorithm Election. Applied Sciences, 13(16), 9125.