iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VII **Month of publication:** July 2023

DOI: https://doi.org/10.22214/ijraset.2023.54751

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VII Jul 2023- Available at www.ijraset.com

A Study of Performance Measures and Throughput of Raft Consensus Algorithm

Stuti Vora[1], Nidhi Thakkar[2], Ravi Gor[3]
[1]*Research Scholar, Department of Applied Mathematical Science, Actuarial Science and Analytics, Gujarat University*
[2]*P.G. Student, Department of Applied Mathematical Science, Actuarial Science and Analytics, Gujarat University*
[3]*Department of Applied Mathematical Science, Actuarial Science and Analytics, Gujarat University*

Abstract: ***Due to its decentralised structure, blockchain technology has acquired great traction in fields such as banking, business, healthcare, and e-voting due to its decentralised nature. Because of the difficulty of lesser throughput, many blockchain systems have sought efficiency enhancements. Mathematical models such as Markov processes, queuing theory, and game theory are employed to analyze various aspects of blockchain systems, including mining processes, consensus mechanisms, and overall performance. This paper focuses on using M/M/1 queuing theory to analyze blockchain mining processes, utilizing the Raft algorithm for consensus. A simulation done by using python is also provided.***
Keywords: Blockchain, Throughput, Consensus, Queueing theory, Raft consensus

I. INTRODUCTION

The successful launch of Bitcoin in 2010 brought blockchain into the spotlight. Numerous companies are adopting blockchain technology including software, financial services, healthcare, and banking. Blockchain is a peer-to-peer distributed database and transaction system that is decentralized, secure, and openly accessible. Its security and performance are largely dependent on consensus algorithm. Consensus algorithms are essential for maintaining the performance and security of blockchain networks. Practical Byzantine Fault Tolerance (PBFT) and Raft are the traditional consensus algorithms used in private blockchain.

Raft algorithm has been widely used in distributed systems because of its high efficiency and simplicity as compared with PBFT. It utilizes a leader-based consensus process, where the leader manages client requests, duplicates logs, and distributes them to nodes. It allows freshly created blocks to be directly added to the chain without additional consensus procedures. However, Raft can tolerate up to 50% of crash fault nodes but cannot handle malicious nodes. Certified nodes act as participants in private blockchain networks, ensuring the integrity and security of the system.

The factors that influence blockchain performance are.

1) *Transaction Latency:* Transaction latency refers to the time between the time a transaction is requested and finally committed. Transaction latency must be short for a blockchain to achieve high performance.
2) *Transaction Throughput:* Transaction throughput represents the rate at which a valid transaction is processed within a set period of time and is referred to as TPS.

II. LITERATURE REVIEW

Memon et. al. [5] (2019) analyzed a model based on queuing theory that replicated a blockchain. The M/M/1 queue was used for the mining pool, while the M/M/c queue was used as the memory pool, in the provided model. This model was a straightforward but efficient method to show a variety of significant measurements, including (a) The number of client requests per block (b) The mining time of each block (c) System Throughput (d) Memory-pool count (e) Waiting Time in memory-pool (f) The number of unconfirmed client requests in the whole System (g) Total number of client requests and (h) number of blocks generated.

Kim et. al. [3] (2021) proposed a mechanism for electing leaders via the Raft algorithm considering network stability. Developed method reduced the probability of network split using Raft. It also prevented further performance degradation when a blockchain node network is unstable.

Tomić [7] (2021) studied functional advantages and disadvantages of observed protocol. Characteristics compared: security, trust among participants, throughput, and scalability. They concluded that no protocol showed complete dominance in all aspects of the comparison. Most protocols in practice show deviations from the theoretically stated optimal results. All the observed proto - cols like PBFT, FBA, DBFT were intended for permissioned blockchains, not all have the same purpose. So, when choosing a consensus protocol for blockchain applications, take into purpose priority.

Vora and Gor [8] (2022) used blockchain queueing theory to analyze theoretical transaction-confirmation time of a block. The blockchain system was examined at arrival as well as departure point. This paper reflects on M/G/1 queue and examines the blockchain system using queueing theory.

Yang [10] (2022) performed a deep analysis of the consensus techniques for running BaaS in clouds. Experimented with variations on the number of orderer nodes, batch sizes, payload sizes, and fault nodes. Based on the experiment results, they provide key findings on 1) resource consumption of the three consensus algorithms, 2) the effect of block size and block creation rate on performance, and 3) the existence of high and low orderer nodes in terms of CPU cycles. That study contributed to understanding the characteristics of resource consumption.

Vora and Gor [9] (2022) provided a markovian queueing model of the form $M^b/M/1$ as the theoretical foundation for creating a Proof of Authority (PoA)-based blockchain system. It was focused on the stochastic behavior of the client requests, consensus process, and throughput. Client requests were thought to arrive in batches with a Poisson distribution, according to the model. The queueing model had been used to analyze the PoA consensus protocol. To demonstrate the reliability of the underlying simulation model, numerical experiments carried out in Python were presented. It had been noted that the block's optimal throughput rate was attained for small numbers of client requests.

III. RAFT CONSENSUS ALGORITHM [2],[3]

Consensus means multiple servers agreeing on the same information. The Raft consensus algorithm is a distributed consensus algorithm that verifies that few nodes in a distributed system agree on a system's condition.

It is used in permissioned blockchain. In permissioned blockchain participants must have an invitation or permission to participate. Raft consensus algorithm was introduced in 2014 by Diego Ongaro and John Ousterhout after Reliable, Replicated, Redundant and Fault-Tolerant.

1) *Leader Election:* In Raft, there is always one leader that coordinates the actions of the nodes in the cluster. The first step is to elect a leader. The nodes in the cluster send out "heartbeat" messages to each other. If a node does not receive a heartbeat message from the leader for a certain period, it will start an election. Each node will send out a vote request message to all other nodes in the cluster, and the node with the most votes become the leader.
2) *Log Replication:* Once a leader is elected, it begins to accept requests from clients and sends Append Entries messages to the other nodes in the cluster to replicate its log entries. Each log entry contains a command from the client and a unique index. The leader waits for acknowledgements from most nodes before it can commit the log entry.
3) *Commitment:* Once a leader receives acknowledgments from most nodes, it commits the log entry to its own log and sends out a Commit message to all other nodes in the cluster. Upon receiving the Commit message, each node applies the log entry to its state machine, which updates the state of the system.
4) *Recovery:* If the leader fails or loses connectivity, a new leader election is started. In this case, the new leader will search for the most up-to-date log entry and replicate it to all nodes that have outdated logs.

A. *Election Process*

In the Raft consensus algorithm, the nodes (i.e., server computers) vote for a leader, and the other nodes then follow the leader. Considering that every node is trustworthy and does not have any malicious intentions. When a timeout occurs, the system elects a new leader because there is no one now. During the election process few candidates compete for the role of leader and the rest of the nodes choose the winner.

Each node has a consensus module, which is always operating within one of the following modes:

1) *Follower:* A passive node, which only responds to RPC's and will not initiate any communications.
2) *Candidate:* An active node which is attempting to become a leader, they initiate Request Vote RPC's.
3) *Leader:* An active node which is currently leading the cluster, this node handles requests from clients to interact with the replicated state machine and initiates Append Entries RPC's.
4) *Voting Request:* The candidate who receives the most votes become the leader. The followers can vote in support of or against the leader's proposition. The leader candidate requests the votes. This voting request contains two parameters:
5) *Term:* The last calculated number known to candidate + 1.
6) *Index:* Committed transactions available to the candidate.

These methods perform in numerous rounds, and the term indicates a new voting cycle. If the last round of voting is completed, the next term will be the old term number Plus 1. The index displays the committed transactions that are candidates but not yet committed.

B. Making Decisions as a Follower Node

After receiving a voting request, the nodes' responsibility is to give their vote for or against the candidate. So, in the Raft consensus approach, a leader is chosen in this manner. Each node compares the term and index that it has received to the corresponding, currently known values. The voting request is received by the node. It compares the previously viewed term to the newly received term. Because the node views this request as an old request, if a newly received term is less than the already observed term, it discards it. The newly received term is greater than the previously seen term. It compares the recently obtained index number to the one that has already been observed. It votes for the candidate if the recently obtained index number is higher than the one already observed; else, it stays away.

C. Leader Ordered Node Selection [10]

The candidates that receive the majority of the nodes' votes are declared the leaders, while the remaining nodes are made up of their followers. Through a "heartbeat mechanism," Raft maintains at least one leader node. The leader node sends "heartbeats" on a regular basis to alert the following ordered nodes about its presence. The leader election is started by the follower ordered nodes if they do not get the heartbeat by a specific time. When a new leader ordered node is elected, it begins sending heartbeats to the follower ordered nodes and generating blocks.

IV. MATHEMATICAL MODEL

This section describes the mathematical queuing approach of raft consensus algorithm. It determines the client request arrival, leader election process and block building process in blockchain. Here, M/M/1 queueing model is considered for this raft consensus system, in which arrivals are modelled as a Poisson process and service times are exponentially distributed. Client request arrives in the system at rate λ. $1/\mu$ is the required service time taken by the leader node as a server. Here, the election timeout is not considered.

1) *Arrival Process:* The client requests are arrived at rate λ in the network of nodes. The client requests are accumulated in the FCFS[7] fashion.
2) *Service Process:* When client requests come into the system, it is first checked whether there is a leader or not. If there is no leader, then the leader's election is processed, and the leader is elected. The new leader sends a heartbeat message to all follower nodes.

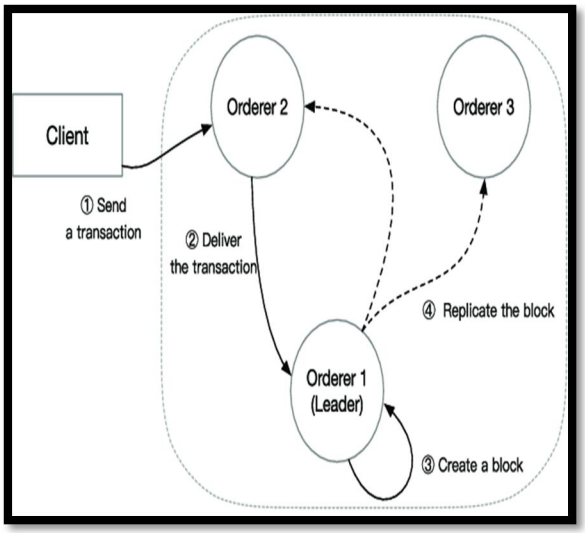


Figure 1. Raft Consensus Algorithm [10]

After the election process, leader node bundled those requests in block. A bundled block of client requests is sent by the leader node to the follower node [7]. After receiving the block, each follower node stores the received block. The leader node then sends messages to the follower nodes to confirm whether all follower nodes have successfully received the block [7]. If they have not, the leader retransmits the block. When all ordered nodes have successfully replicated the block, the ordered nodes (including leader and follower) distribute the block to the peer nodes. Table 1 shows the terminology used in this paper.

A. *Terminology*

Table 1: Terminology	
λ	Arrival rate of client request
μ	Service rate of client request
ρ	Utility factor of Blockchain system
$N(t)$	Total Number of client request in the system at time t
$N_q(t)$	Number of client request in queue at time t
$N_s(t)$	Number of client request in system at time t
T_q	Time a transaction spends to wait in a queue
T	Total time a transaction spends in the system
$E[N]$	Average number of client request in system
$E[N_q]$	Average number of client request in queue
$E[T_q]$	Average waiting time in queue
$E[T]$	Average waiting time in the system
$E[S]$	Average Service time

V. MATHEMATICAL FORMULATION

Here, a single-server M/M/1 queue in steady state is considered. Interarrival times and service times are assumed to be exponentially distributed with density functions,

$$a(t) = \lambda e^{-\lambda t},$$
$$b(t) = \mu e^{-\mu t}$$

Thus, $\lambda_n = \lambda\ (n \geq 0)$ and $\mu_n = \mu\ (n \geq 1)$

The flow-balance equations for this system are,

$$(\lambda + \mu)p_n = \mu p_{n+1} + \lambda p_{n-1} \quad (n \geq 1),$$
$$\lambda p_0 = \mu p_1$$
$$p_1 = \frac{\lambda}{\mu} p_0$$

Solving $\{\ \mathbf{p_n}\ \}$ using an iterative method,

$$\text{Probability, } p_n = p_0 \left(\frac{\lambda}{\mu}\right)^n, (n \geq 1) \qquad \ldots (1)$$

Now, $\sum_{n=0}^{\infty} P_n = 1$

$$1 = \sum_{n=0}^{\infty} p_0 \left(\frac{\lambda}{\mu}\right)^n = p_0 \sum_{n=0}^{\infty} \rho^n$$

Here, $\rho = \lambda/\mu$, where ρ is the utility factor or traffic intensity (i.e., Probability that all the servers are in use).

So, $$p_0 = \frac{1}{\sum_{n=0}^{\infty} \rho^n} \qquad \ldots(2)$$

$\sum_{n=0}^{\infty} \rho^n$ is a geometric series that converges if and only if $\rho < 1$.

$$\sum_{n=0}^{\infty} \rho^n = \frac{1}{1-\rho}\ (\rho < 1), \qquad \ldots. (3)$$

Using equation (3), equation (2) becomes

$$p_0 = 1 - \rho\ \ (\rho = \lambda/\mu < 1) \qquad \ldots(4)$$

Hence, equation (1) becomes,

$$p_n = (1 - \rho)\, \rho^n\ \ (\rho = \lambda/\mu < 1) \qquad \ldots (5)$$

A. *Performance Measures*

1) *Average number of client request in the system $E[N]$*

$$E[N] = \sum_{n=0}^{\infty} n p_n$$
$$= (1-\rho) \sum_{n=0}^{\infty} n\rho^n$$
$$= \rho \sum_{n=1}^{\infty} n\rho^{n-1}$$

$\rho \sum_{n=1}^{\infty} n\rho^{n-1}$ is simply the derivative of $\sum_{n=0}^{\infty} \rho^n$ with respect to ρ ,

$$\sum_{n=1}^{\infty} n\rho^{n-1} = \frac{d[1/(1-\rho)]}{d\rho}$$
$$= \frac{1}{(1-\rho)^2}$$

So, the expected number of transactions in the system at steady state is,

$$E[N] = \frac{\rho(1-\rho)}{(1-\rho)^2}$$
$$= \frac{\rho}{(1-\rho)}$$
$$E[N] = \frac{\lambda}{(\mu - \lambda)} \quad \ldots(6)$$

2) Average number of of client request in the queue $E[N_q]$

$$E[N_q] = \sum_{n=1}^{\infty}(n-1)p_n$$
$$= \sum_{n=1}^{\infty} np_n = \sum_{n=1}^{\infty} p_n$$
$$= E[N_s](1-p_0)$$
$$= \frac{\lambda^2}{\mu(\mu-\lambda)} \quad \ldots(7)$$

3) Average confirmation time of client request in system using Little's result $E[N] = \lambda E[T]$

$$E[T] = \frac{E[N]}{\lambda} = \frac{1}{(\mu-\lambda)} \quad \ldots(8)$$

4) Average waiting time in the queue $E[T_q]$ using Little's result $E[N_q] = \lambda E[T_q]$

$$E[T_q] = \frac{E[N_q]}{\lambda} = \frac{\rho}{(\mu-\lambda)} \quad \ldots(9)$$

In particular, Eq^n (4) is valid only when $\rho < 1$. $\rho \geq 1$ get a negative number which means the system and queue becomes unstable.

5) Throughput

Throughput is an important metric for blockchain systems as it measures the number of client requests that can be processed by the network within a given time. The throughput of a blockchain platform is affected by several factors such as block size, block time, network latency, and consensus algorithm. The Raft consensus algorithm can vary depending on several factors such as the size of the cluster, the network conditions, and the workload. Throughput of a Raft consensus algorithm can be given by:

$$Throughput = \frac{Total\ number\ of\ committed\ entries}{Time\ taken\ to\ commit\ them} \quad \ldots(10)$$

VI. NUMERICAL EXPERIMENTS AND RESULT

Performance measure of Raft consensus process in Blockchain system is analysed numericaly. Numerical experiments for arrival rates $\lambda = 0.3, 0.5$ and service rate $\mu \in [0.5, 1.0]$ is performed using Python.

μ	$E[T_q]$	$E[T]$	$E[N_q]$	$E[N]$	$Throughput$
0.589	4.469	4.483	1.052	1.158	0.2
0.621	2.841	2.857	0.894	0.926	0.2
0.656	1.957	1.964	0.756	0.782	0.2
0.709	1.438	1.368	0.614	0.635	0.21
0.748	1.19	1.205	0.57	0.572	0.22
0.81	0.894	0.896	0.512	0.514	0.23
0.866	0.588	0.6	0.444	0.449	0.24
0.906	0.394	0.4	0.392	0.401	0.25
0.95	0.192	0.2	0.309	0.317	0.27
0.964	0.000	0.000	0.251	0.295	0.4

$Table\ 2 : Value\ for\ \lambda = 0.3$

Figure 2(a) and 2(b) shows the average client requests confirmation time in system as well as in queue. As service rate increases, average confirmation time of client requests decreases. Figure 2(c) and 2(d) shows the average number of client requests in system as well as in queue. As service rate increases, the average number of client requests decreases.

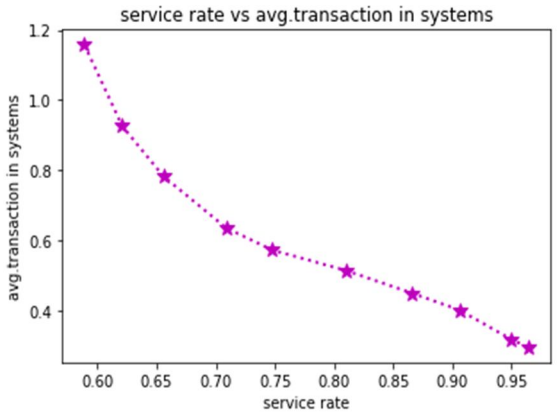


Figure 2(c). Graph for Avg.no of transactions in system

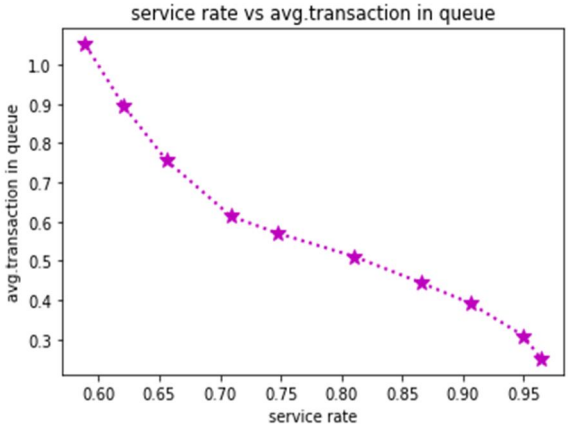


Figure 2(d). Graph for Avg.no of transactions in queue

Table 3 : Value for λ = 0.5

μ	$E[T_q]$	$E[T]$	$E[N_q]$	$E[N]$	$Throughput$
0.589	3.545	3.557	1.025	1.041	0.1
0.621	2.401	2.42	0.889	0.902	0.11
0.656	1.891	2.018	0.798	0.82	0.12
0.709	1.413	1.438	0.706	0.734	0.14
0.748	1.262	1.281	0.672	0.69	0.15
0.81	1.149	1.167	0.645	0.661	0.16
0.866	0.956	0.987	0.571	0.597	0.17
0.906	0.625	0.649	0.499	0.507	0.18
0.95	0.301	0.341	0.42	0.472	0.2
0.964	0.000	0.000	0.396	0.407	0.4

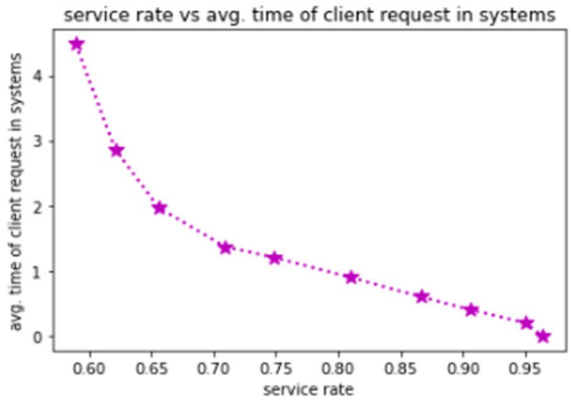


Figure 2(a) Graph for Avg. Confirmation time of client request in system

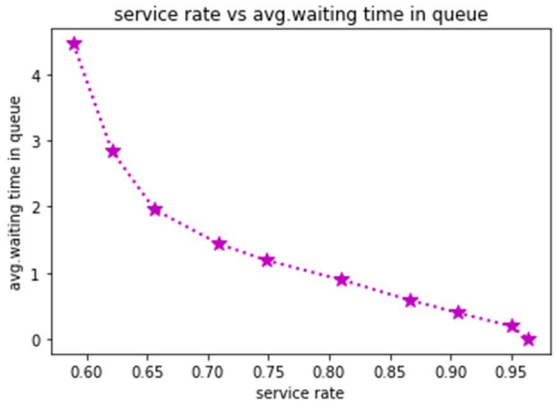


Figure 2(b) Graph for Avg. waiting time in queue

Figure 3(a) and 3(b) shows the average client requests confirmation time in system as well as in queue. As service rate increases, average confirmation time of client requests decreases. Figure 3(c) and 3(d) shows the average number of client requests in system as well as in queue. As service rate increases, the average number of client requests decreases.

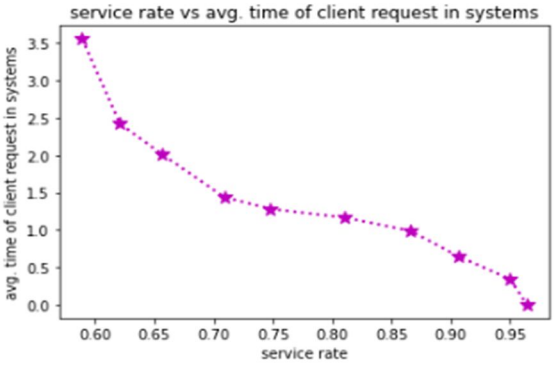


Figure 3(a) Graph for Avg. time of client request in system

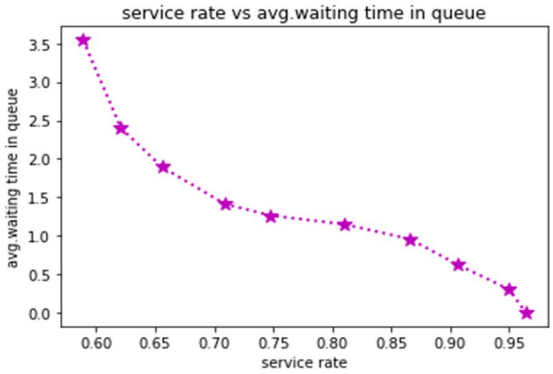


Figure 3(b) Graph for Avg. waiting time in queue

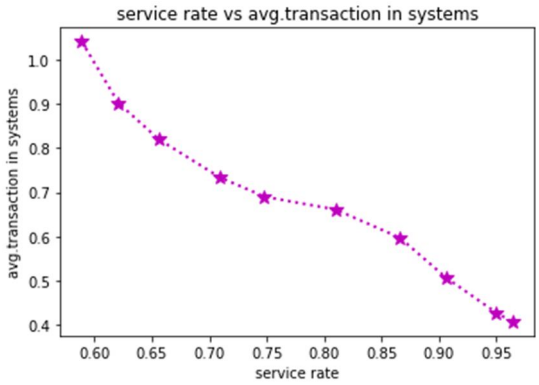


Figure 3(c). Graph for Avg.no of transactions in system

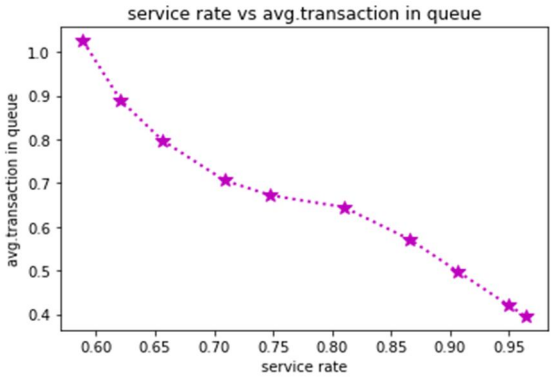


Figure 3(d). Graph for Avg.no of transactions in queue

For arrival rate $\lambda = 0.3$, figure 4(a) shows as service rate increases throughput decreases and same for figure 4(b) for arrival rate $\lambda = 0.5$.

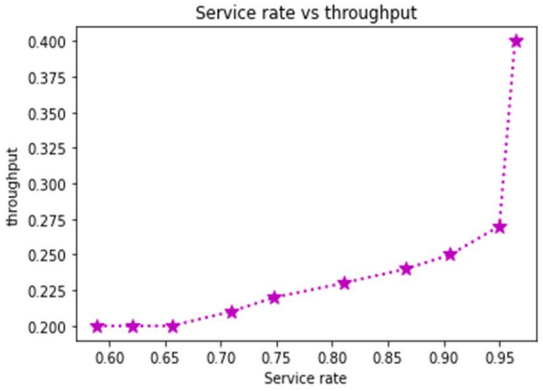


Figure 4(a). Service rate vs Throughput for $\lambda = 0.3$

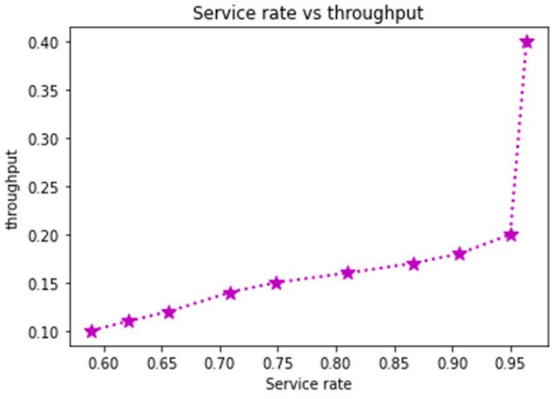


Figure 4(b). Service rate vs Throughput for $\lambda = 0.5$

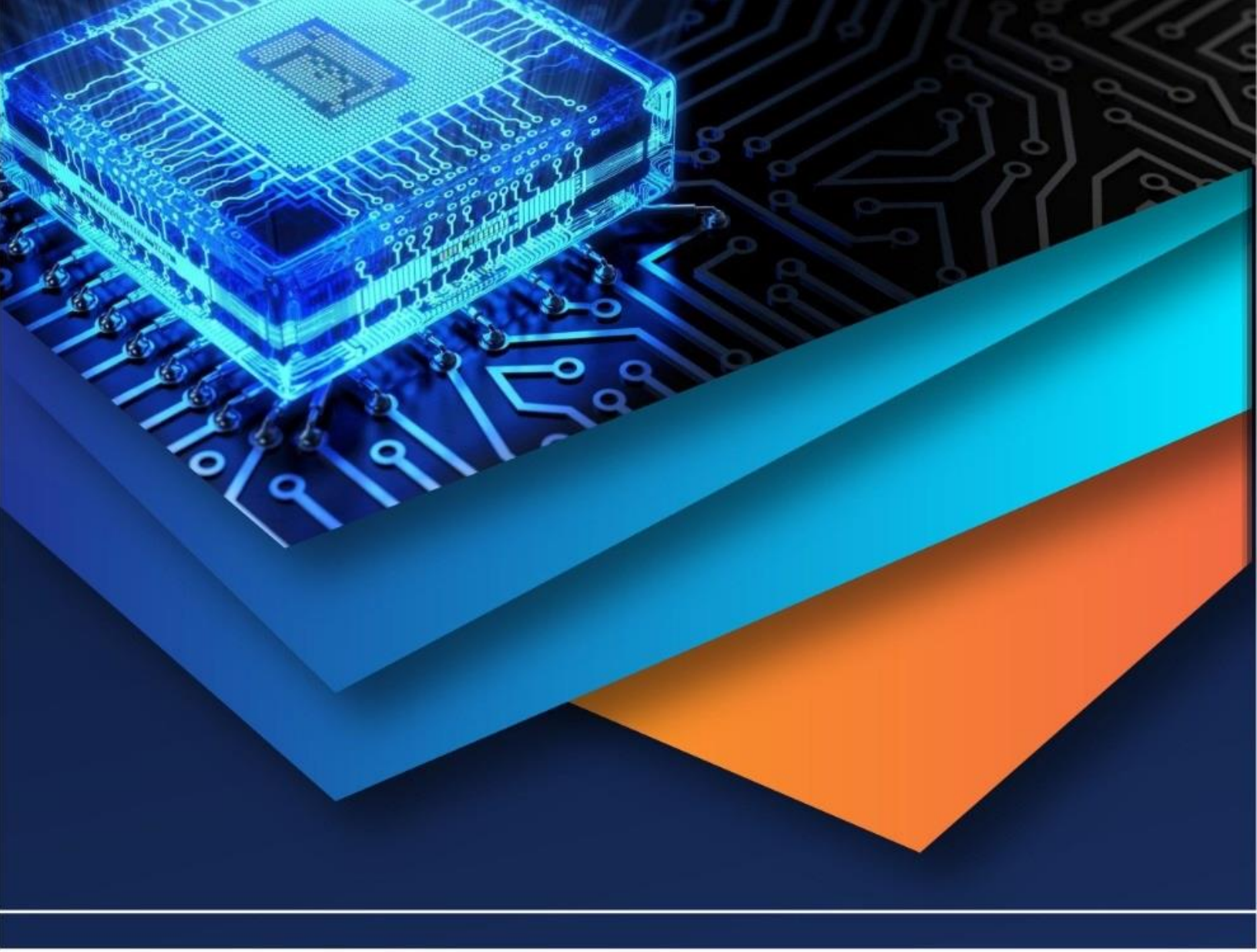
VII. CONCLUSION

This study presents a theoretical examination of the Raft consensus process using M/M/1 queuing model for blockchain. The proposed methodology is straightforward yet effective to display numerous different parameters, such as average number of client request, average waiting time in the system as well as in queue. The analysed results demonstrates that as the service rate increases, throughput increases, and the other primary performance measures decreases.

Numerical experiments and results are performed in python. This study opens several future research directions for optimizing the performance of the Raft consensus algorithm, considering scalability, network characteristics, empirical validation, comparative analysis with other algorithms, and exploring different workload scenarios. These approaches can contribute to the development of efficient and robust consensus algorithms for blockchain systems.

REFERENCES

[1] Baliga, A., Subhod, I., Kamat, P., & Chatterjee, S. (2018). "Performance evaluation of the quorum blockchain platform". arXiv preprint arXiv:1809.03421.
[2] Huang, D., Ma, X., & Zhang, S. (2019). "Performance analysis of the raft consensus algorithm for private blockchains". IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50(1), 172-181.
[3] Kim, D., Doh, I., & Chae, K. (2021, January). "Improved raft algorithm exploiting federated learning for private blockchain performance enhancement". In 2021 International Conference on Information Networking (ICOIN) (pp. 828-832). IEEE.
[4] Li, Q. L., Ma, J. Y., & Chang, Y. X. (2018). "Blockchain queue theory". In Computational Data and Social Networks: 7th International Conference, CSoNet 2018, Shanghai, China, December 18–20, 2018, Proceedings 7 (pp. 25-40). Springer International Publishing.
[5] Memon, R.A.; Li, J.P.; Ahmed, J. "Simulation Model for Blockchain Systems Using Queuing Theory". Electronics 2019, 8, 234. https://doi.org/10.3390/electronics802023004
[6] Ongaro, D., & Ousterhout, J. (2014). "In search of an understandable consensus algorithm". In 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14) (pp. 305-319).
[7] Tomić, N. Z. (2021). "A review of consensus protocols in permissioned blockchains". Journal of Computer Science Research, 3(2), 32-39.
[8] Vora S.& Gor R. (2022), "Study of Average Transaction Confirmation time in a Blockchain using Queueing Model", IOSR Journal of Computer Engineering (IOSR-JCE),24(3), 2022, pp. 60-68.
[9] Vora and Gor, "Queueing Model for PoA based Blockchain system" IOSR Journal of Mathematics (IOSR-JM).
[10] Yang, G., Lee, K., Lee, K., Yoo, Y., Lee, H., & Yoo, C. (2022). "Resource Analysis of Blockchain Consensus Algorithms in Hyperledger Fabric". IEEE Access, 10, 74902-74920.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY