IJECBE (2023), **1**, **1**, 44–56
Received (17 July 2023) / Revised ()
Accepted (23 July 2023) / Published (29 September 2023)
https://doi.org/10.62146/ijecbe.v1i1.20
https://ijecbe.ui.ac.id
ISSN 3026-5258

International Journal of Electrical, Computer and Biomedical Engineering

RESEARCH ARTICLE

Performance Evaluation of QUIC Protocol in Message Replication Overhead in PBFT Consensus using NS-3

Thio Lutfi Habibi and Riri Fitri Sari*

Department of Electrical Engineering, University of Indonesia, Depok, Indonesia
*Corresponding author. Email: riri@ui.ac.id

Abstract

The development of protocols in the ICT world to increase reliability and speed in data traffic gave an inspiration to a new protocol called the QUIC protocol. The QUIC protocol is expected to improve the performance of Transport Control Protocol (TCP). In addition, developments also occur in blockchain technology where the protocol used in this technology still uses the existing TCP protocol. In this paper, we aimed to research whether the QUIC protocol implementation in blockchain infrastructure could improve the performance of the blockchain infrastructure itself, in terms of the time required for transactions. We focus on conducting research to measure the overhead time reduction of Practical Byzantine Fault Tolerance (PBFT) consensus by implementing the QUIC protocol, the consensus propagation process is a crucial phase in Blockchain. For simulation we used NS-3 discrete simulation environment to conduct scenario and our simulation result showed that the QUIC Protocol have potential significant performance compared to TCP Protocol in large datasets, on the other hand QUIC protocols have more room for improvement by implementing appropriate congestion algorithms.

Keywords: QUIC, Transport Protocol, NS-3, Blockchain, Consensus, PBFT

1. Introduction

In the current era of ICT development, protocols play a major role for data exchange and communication for the regulation and standard. Protocol development has been through so many advancements and discovery of several new protocols [1], in improving the performance of the existing protocols. One of the new protocols being developed is the QUIC protocol. This protocol was initially developed by the leading search engine company, Google. Currently, the QUIC protocol has been adapted by the IETF in May 2021. The purpose of developing this protocol is to reduce latency and overhead when using the TCP protocol. Using TCP protocol, we need to establish a connection with the "three-way handshake". Three-way handshake involves three steps, the client sends a SYN message, the server sends a message that combines an ACK for the client's SYN and contains the server's SYN; and then the client sends an ACK for the server's SYN.

The implementation of the QUIC protocol is expected to improve the performance of applications, especially those using this protocol stack. QUIC is also very helpful in high round-trip time conditions [2]. The QUIC protocol is implemented on top of the UDP protocol so that it can be used for compatibility on the current internet infrastructure, however with the advantages of the QUIC protocol, thus can improved performance.

On the other hand, another technology that is currently taking place is blockchain. Blockchain is a technology introduced by Satoshi Nakamoto, in his whitepaper entitled peer-to-peer electronic cash system in 2008 [3]. Basically, blockchain is a distributed system that is interconnected to be able to jointly carry out a certain task which refers to a single ledger basis already defined. Blockchain itself still uses the underlying protocol commonly used, namely TCP. In this study, we tried to integrate the advantages of the new protocol, namely the QUIC protocol to get an idea of whether this protocol can improve the performance of the blockchain itself.

Since blockchain is a system consisting of multiple nodes which are linked together and distributed fragments of data among them, the consensus algorithm becomes a crucial part of building data transactions in blockchain. The consensus algorithm manages and validates transaction data in the blockchain against all participating nodes in the system. A transaction will not become a commit state until the consensus algorithm declares data validity across all nodes, then consensus and network layers are the core of a blockchain platform and terms of cost/complexity needs to be analyzed before additional layers are integrated [4]. The consensus technique known as Practical Byzantine Fault Tolerance (PBFT) is built to function via a series of agreement communication rounds between a minimum set of 3f+1 nodes for each f problematic nodes. To achieve consensus, PBFT relies on a strong message-passing protocol among the copies. In PBFT, the replicas exchange many messages during each communication round. A three-way handshake will be expensive overhead in PBFT consensus; the number of communications will be directly proportional to the number of replicas.

In general, the comparison between TCP and UDP will lead to a comparison of reliability vs. performance, where TCP, as a connection-oriented protocol, will coordinate data transmission through a three-way handshake, which will be an overhead cost for communication performance. At this point, UDP, as a connectionless protocol, has an advantage in terms of performance since communication is handled individually and can be forwarded to the destination without any prior coordination. In addition, in terms of protocol security, Datagram Transport Layer Security (DTLS) inherits the nature of UDP, which is connectionless and allows for flooding encrypted traffic, compared to TLS, which is used on the TCP protocol via SNI Hello then

Secure communication can be ensured. The QUIC protocol, which utilizes the UDP transport layer but inherits the connection-oriented nature of TCP, is anticipated to offer new insights into distributed system infrastructure and transport protocols, particularly in the context of consensus algorithms, where reliability, performance, and secure communication requirements are essential. In this study, we will assess the QUIC protocol's performance features to implement it for PBFT consensus execution. As the basis for implementing the QUIC protocol, TCP and UDP will be compared in this instance.

Furthermore, the organization of this research in this paper is as follows. Section II explains the references from literature review from previous. Section III describes the simulation of the QUIC protocol which is integrated into the routine of the blockchain using the NS-3 simulation tools. Section IV explains the results of the research and simulations carried out in the previous section. The last, Section V which contains the conclusions of this study.

2. Study Literature

2.1 QUIC Protocol

The QUIC protocol is a transport protocol originally designed by Google and implemented at the application level. The development of the QUIC protocol is expected to overcome some problems in the transport layer which is commonly used in TCP protocols. From the point of view of recentness of this protocol, we can highlight Important features of QUIC [5]:

1. Communication Implementation Latency: The QUIC protocol combines encryption with handshake transit to reduce the number of communication round trips. It gives you cached client code that you can use to talk with the server you want. This eliminates the requirement for a fresh handshake.
2. Multiplexing: QUIC multicast is made up of streams that each carry their own data. The stream identification ID is used to send data for each stream in the specified frame. One or more frames can be included in a QUIC package.
3. Forward Error Correction: FEC is supported by QUIC, with each FEC packet containing the parity of the packets that make up the FEC group. This feature can be turned on or off as needed. It enables him to retrieve the contents of a packet that has been lost in an FEC group.
4. Connection Migration: Instead of the 4-word set of source and destination IP addresses and crucial communication port numbers, QUIC connections are identified by a 64-bit connection identifier.

De Biasio et al, in [6] replicated The QUIC protocol into discrete simulation tools NS-3 that we will explain in section II.C in urge demand of RFC submission of QUIC Protocol into IETF as HTTP/3 implementation.

The research conducted by extended TCP module with features that assimilate QUIC in terms of stream multiplexing, low-latency initial handshake, improved SACK through ACK frames. QUIC Protocol is tested by measuring the Round Time Trip (RTT) TCP congestion control such as New Reno, Vegas, and Look like New Reno with additional flavor of QUIC Protocol.

2.2 Practical Byzantine Fault Tolerance (PBFT)

The Practical Byzantine Fault Tolerance (PBFT) is a consensus algorithm that was originally developed as a mechanism for ensuring the integrity of distributed networks that was introduced by Castro and Liskov [7]. To add the next block, this algorithm requires that all nodes participate in the voting process with minimum of 3f+1 nodes participated for anticipated f nodes of failure. A two-thirds majority or minimum of 2f+1 vote is required as a quorum to reach an agreement. PBFT has three important components: view, primary, and replica. Replica nodes are used to ensure the effectiveness of the voting process. To ensure network integrity, the PBFT algorithm sends and receives a lot of messages between nodes [8]. So far, many PBFT patches have been developed. Delegated BFT is not the same as PBFT. For example, not all nodes, but only some delegates, participate in the voting process. Validators construct and propose new transaction blocks using Simplified BFT, a Byzantine fault tolerance technique.

PBFT works with five phases, there is request, pre-prepare, prepare, commit and reply [9]. Figure 1 describes how PBFT works in case 4 nodes to tolerate 1 failure. When request phase sent by Client to the primary node, then it forwards messages to the other three nodes. If node 3 fails, a single message is sent to all the nodes in a five-step process to obtain a consensus. Finally, to finish the consensus round in commit phase each node needs minimum 2f+1 vote to agree the consensus then send reply message to client. In each consensus round, PBFT assures that nodes maintain a common state and conduct consistent actions. The PBFT protocol is a consensus protocol because it achieves a high level of consistency. [4, 9].

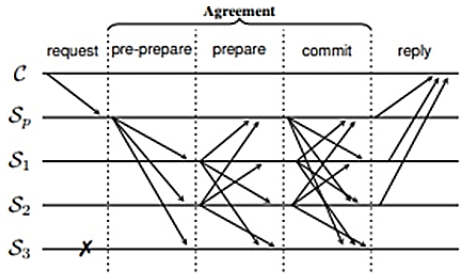

Figure 1. Single-layer PBFT consensus processing [7]

2.3 NS-3

NS-3 [10] is a simulator device used to simulate net systems, wi-fi networks, etc. The competencies of this simulator device are constructed at the GNU/Linux system. Some of the stairs utilized in NS-3 are topology definition, version building, hyperlink and node configuration, execution, overall performance analysis, and photo visualization. NS-3 works as discrete simulator that represents system behavior as a series of (discrete) temporal events. Each event occurs at a specific point in time and indicates a change in the status of the system [6]. No changes are expected to occur in the system between consecutive events. As a result, the simulation time can skip straight to the time of the next occurrence. The next event is referred to as the passage of time [7].

Some of the benefits of the NS-3 in comparison to the preceding model is that the NS-3 is extra documented than the NS-2 and is continually up to date concerning tendencies that arise in new technology and NS-3 community actively conducted paper submission about new protocol or RFC simulation then it encourages NS-3 have more module that can represent new technologies in real world problem. This simulator device is modular, in step with actual conditions, and helps integration with virtual environment.

To measure or know the performance of a protocol, a simulation tool is needed. One of the widely used network simulation tools is NS-3 [10]. NS-3 or Network Simulator 3 is a discrete event simulator to perform simulations on protocols or network infrastructure that currently exists and is usually used in the scope of research and learning by performing how packet and data network works [11], NS-3 is also an open-source platform which means that the source code can be downloaded freely and can be modified by the user. NS-3 itself has quite a capability, it supports many network protocols currently available. The NS-3 is also equipped with many modules which can be further developed by contributors. This simulator also allows running simulations in parallel and in a distributed manner.

2.4 Blockchain Consensus Simulator

Zhayujie, has developed simulations of blockchain consensuses namely PBFT, Raft, and Paxos using UDP protocol based on NS-3 version 3.29 [12]. This project tries to model the relationship between blockchain performance and several aspects such as network scale, block size, bandwidth, delay, and others. In this research we adopted flow works simulation of PBFT consensus and modified it to be ready implemented with any other protocol with TCP and QUIC.

3. QUIC Protocol and Blockchain: An Overview

With the submission of the QUIC Protocol into the IETF proposal as an HTTP/3 implementation, the need for Application Layer development simulations is increasing because the position of the QUIC Protocol working at the transport layer will affect the layer above it. In a simulation study of the QUIC protocol using NS-3 [6], the implementation of the QUIC protocol is based on RFC revision 113, which shows an increase in performance in the transport layer. In this study, we make substitute to the TCP Socket Protocol to meet the characteristics of the QUIC protocol, namely utilizing UDP Datagram transport which has TCP reliability capabilities such as congestion control and reduction of the initial handshake delay. This research will help open the door to development on other application layers for testing and simulation development.

Blockchain as a technology that utilizes a linked list of nodes that are integrated with each other to form a large data block is based on a certain consensus. In the development of Blockchain transactions, especially at the message propagation stage through the consensus method. Transaction performance development research mostly concentrates on developing consensus methods, in some research method development, especially research [13] on development plans also consider the implementation of the QUIC protocol, however this idea is not proceeded because Blockchain is peer-to-peer communication while QUIC do not have special treatment for peer-to-peer communication.

Practical Byzantine Fault Tolerance (PBFT) [14] is a consensus algorithm in the blockchain for use in enterprise consortiums where some members or nodes on the blockchain are made trusted. PBFT uses a three-phase commit protocol set by the ledger to approve incoming requests. With three-phase commit and replication, the PBFT is more redundant in terms of communication, and the elements of the PBFT phase are divided into phases: request, pre-prepare, prepare, commit, and respond with a minimum total message count replica of 1 + 3f + 3f(3f-f) + (3f-f+1)(3f+1) + 3f-1 [9]. So, to run PBFT consensus with a maximum of 2 node failures, a minimum message communication of 71 total messages for 1 request will be required.

While PBFT performed well in terms of latency, resource requirements, and node complexity, node scalability, the PBFT bottleneck is node scalability, a statistic that measures how effectively a network represents a system's ability to accommodate an increasing number of nodes [8] because many relies on inter-site communication [15]. This is a motivational research question for the implementation of the QUIC protocol that utilized UDP Datagram above TCP reliability and congestion control in the Blockchain environment since PBFT algorithm intensively exchanges messages between nodes to ensure network integrity [8].

The problem of protocol agreement in PBFT has attracted attention in research [9] which focuses on the problem of PBFT agreement processes that requires replication, especially if the nodes are used in large numbers which of course will increase execution time as overhead time and how to reduce the overhead time. The QUIC protocol is one of the recommended solutions to emphasize the need for overhead time with the assumption that the QUIC Protocol will help speed up the message replication process, however in this study the solution is still a perspective view, no simulation has been carried out to see the role of the QUIC protocol in reducing overhead time.

4. Simulation

4.1 Simulation Environments

The simulation uses a virtual machine with specifications of 6 vCPU and 12GB Memory, using Virtualized Environment Ubuntu 20.04 operating system. NS-3 used is version 3.37. QUIC NS-3 module version 1.0 has been used and was released in November. 13, 2020, without making any module changes.

The NS3 QUIC Protocol module is used as a baseline for the NS-3 blockchain simulator which will later be used as a communication protocol in the consensus PBFT to be simulated. The Socket Class will be a variable to play a role in swinging between UDP, TCP and the QUIC Protocol, other variables that take a part of experiment will be explained in Section IV.B. On the other hand, PBFT Consensus Simulation [12], needs to be adjusted to comply with our experiment since the simulation code designed for UDP Protocol. To implement TCP and QUIC Protocol the simulator must support connection handshake and stream buffer since there are probable the packet needs to be fragmented. The implementation flow of the NS-3 module as illustrated in Figure 2; we integrate both modules into single NS-3 to keep computing performance as a constant variable.

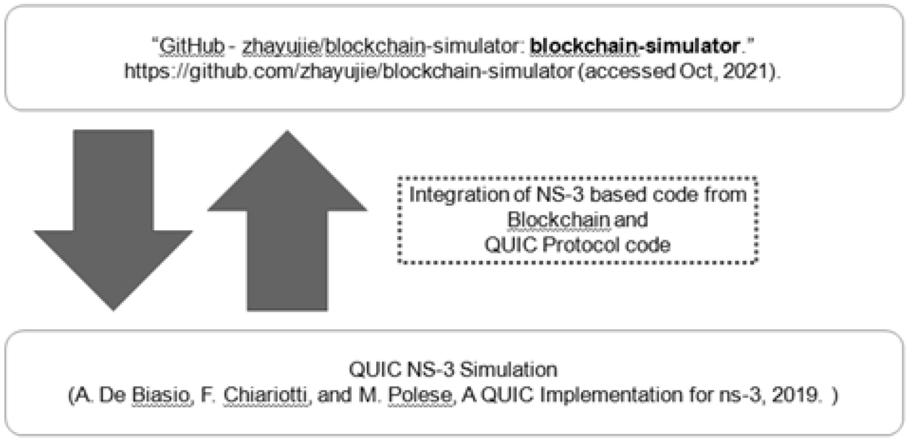
"GitHub - zhayujie/blockchain-simulator: **blockchain-simulator**."
https://github.com/zhayujie/blockchain-simulator (accessed Oct, 2021).
Integration of NS-3 based code from Blockchain and QUIC Protocol code
QUIC NS-3 Simulation
(A. De Biasio, F. Chiariotti, and M. Polese, A QUIC Implementation for ns-3, 2019.)

Figure 2. Flow Simulation Design

4.2 Blockchain Consensus Simulator

To carry out the simulation, it will be implemented with the parameters used for the values that will be implemented in the simulation. Parameters will determine the performance load, parameter values for the simulation are described in Table 1.

Table 1. SIMULATION PARAMETER

Country List	
Parameter	Value
Data Rate	3 Mbps
Delay	3 ms
Nodes	4, 8, 16, 32
Protocol	UDP, TCP, and QUIC
QUIC Congestion Control	Vegas, NewReno, BIC and QUIC Bbr
Size	(Data Rate) * (2*Delay)
N-Iteration	AO & 40
Simulation Tool	NS-3

First, we run the simulation using a blockchain simulator in [12] for the UDP Protocol as a default simulation. The results of the simulation are recorded for their time iteration in PBFT consensus as mentioned in Section III.B. Furthermore, modifications were made to the TCP and QUIC simulation source code by merging the blockchain module in [12] into the module in [6]. Finally, we run a blockchain simulation using the QUIC protocol, and record the iteration result using PBFT consensus.

In the scenario, we conducted simulation using four nodes' sets are 4, 8, 16 and 32 as message propagation node for represent condition up to 10 nodes failure. Delay of 3ms represented as a simulation conducted in LAN with 3 Mbps connection. N-Iteration means number of repetition simulation from first broadcast by initial node until have 40th commit steps of replication in full mesh network as represented in Figure 3. We prepare four congestion control algorithms of QUIC Protocol to measure adjustment that could adjust performance, there are three well known TCP congestion algorithms Vegas, New Reno, and BIC, then the last one is QUIC BBR that as a modified algorithm in NS-3 QUIC Module to comply QUIC behavior. Packet Size we use proportionally between $(DataRate) * (2 * Delay)$ as a dumbbell, that means size will be maximum capacity of bandwidth take round trip for each pair of nodes.

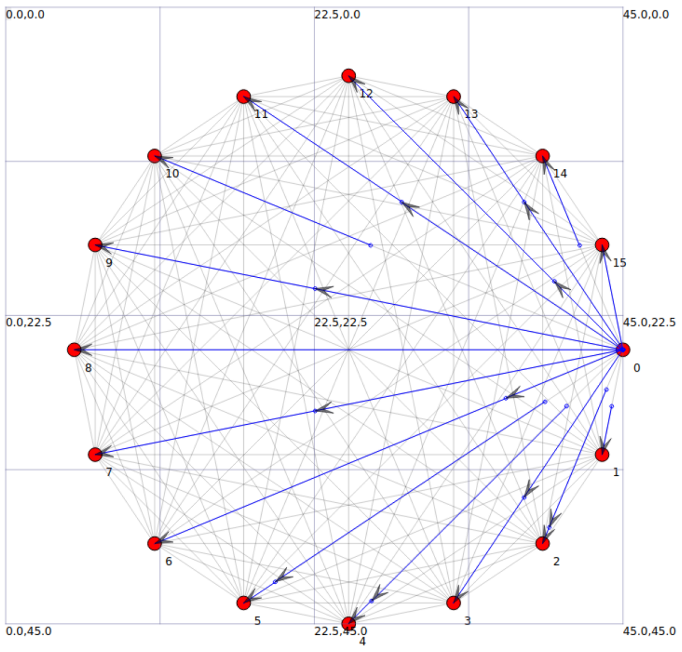

Figure 3. Topology and Flow Communication

As represented in Figure 3, each node will replicate each other in every step of PBFT phase request, pre-prepare, prepare, commit, and respond. In each stage we measure average time from each node to replicate their data and represent them as time execution in that phase. We recorded the time taken to execute the iteration process in the consensus PBFT. The iteration is the replication process carried out by nodes in 1 transaction, after one node receives the call, The node will broadcast it to the adjacent node and the rest of the node will follow until the communication convergent. The total number of iterations follows the formula $1+3f+3f(3f-f)+(3f-f+1)(3f+1)+3f-1$ with f as the number of faults nodes.

The transaction process is divided into NEW ROUND, PRE-PREPARED, PRE-

PARED, COMMIT, COMMITTED, FINAL COMMITTED, and ROUND CHANGE, with New Round and Round Change being the staging step process between iterations, so there are 5 PRE-PREPARED, PREPARED, COMMIT, COMMITTED, and FINAL COMMITTED. Then we recorded the execution time of the first 40 transactions in seconds, representing the 5 main steps in the PBFT. Details of the formulation are discussed in Section III.C.

4.3 Gauges Calculated

In this research, observations, and records of time in each cycle of the consensus PBFT are carried out for each node. Each iteration describes the process of propagation and replication at each node in the consensus PBFT. The time required for each node in the iteration is averaged as a representation of the time required for replication and propagation in each stage, which is described in the following Equation 1 and 2.

$$T = [\overline{t_l}]_{\forall i \in \{1,\dots,n\}} \tag{1}$$

$$\overline{t_l} = \left[\frac{\sum_{j=1}^{m} t_i}{m} \right]_{\forall j \in \{1,\dots,m\};\, m \in \{4,8,16,32\}} \tag{2}$$

Equation 1 represents T as a set value of the average time required for each iteration from 1 to n. The average time required for each n iteration is represented by the amount of time required by each node 1 to m, divided by the number of nodes m described by Equation 2. The representation of time is represented by T for each m set of the number of nodes used in this study, namely 4, 8, 16 and 32.

5. Result and Analysis

Based on simulation we conducted and explained in Section III, we use 4 scenarios with the number of nodes starting from the 4, 8, 16 and 32. We represent the result of the experiment in figures of total time execution for each protocol UDP, TCP and QUIC, average time of each iteration for all protocol and comparison total time execution for each congestion control in QUIC Protocol. Based on the results obtained, generally UDP and TCP have better performance compared to QUIC protocol. UDP still has the best performance compared to TCP, but only a few milliseconds. The performance of the QUIC protocol appears to be affected by the congestion algorithm, where for each algorithm the effect of the execution time is quite different.

Table 2. TOTAL TIME EXECUTION FOR 40th ITERATION (SECONDS)

Protocol	Number of Nodes			
	4	8	16	32
UDP	4.03364	4.13864	4.19464	4.22264
TCP	4.08033	4.163	4.21256	4.24056
QUIC	5.89129	5.89103	5.89087	5.89135

Based on Table 2 and Figure 5 provides an overview of the total time execution each protocol for a given set number of nodes. TCP and UDP have the best performance compared to QUIC Protocol, with total time execution around 4 seconds with difference between TCP and UDP only around 0.03 seconds at most. QUIC total time execution around 5.89 seconds there is a gap time of almost 2 seconds in each set number of nodes.

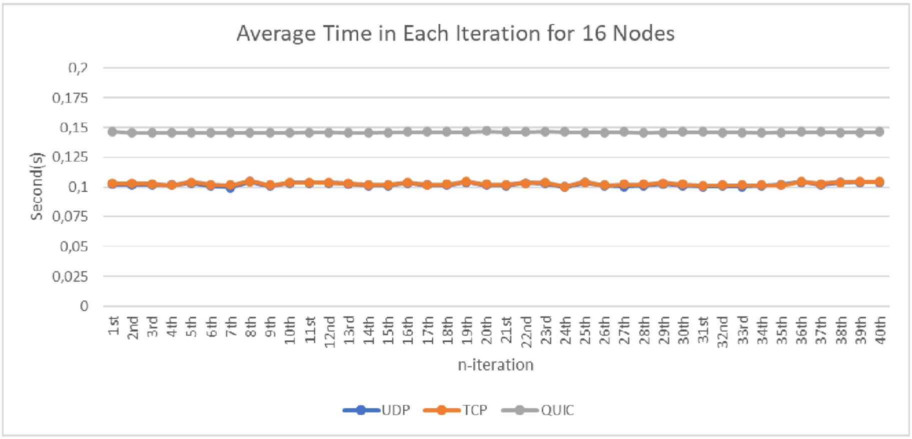

Figure 4. Breakdown of The Average Time for 16 Nodes Case Iteration

From Figure 4, each protocol shows a stable average execution time in each iteration. UDP and TCP do not have any significant differences for average time execution, the average execution time for 16 nodes for the UDP protocol is 0.1022 seconds, for the TCP protocol it is 0.1028 seconds, while for the QUIC protocol it is 0.1460 seconds. Unlike the total time execution with a 2 second gap between QUIC and UDP/TCP, the average time iteration is only 0.04 seconds different.

If we focused on execution time of each set number of nodes, UDP and TCP showed trends that for each set growth around 0.06 to 0.1 seconds. For example, UDP time execution for 4 nodes is 4.033 second and for 8 nodes is 4.138 second there are raising around 100 milliseconds, then 60 milliseconds for 16 nodes, for last set 32 nodes 30 milliseconds. The same pattern happens in TCP Protocol, from 4 to 8 nodes raising around 80 milliseconds then 60 milliseconds and 30 milliseconds for 16 and 32 nodes.

In the other hand, QUIC protocol does not show any significant time execution in every set number of nodes, the increment not more than 0.2 milliseconds. This shows that for UDP and TCP the execution time will be directly proportional to the number of nodes communicating with each other, in the opposite pattern the QUIC protocol shows that changes in the variable number of nodes do not give a significant change, namely at 0.2 milliseconds. If this pattern is continued, then at a very large number of certain nodes there will be a turning point in the comparison of UDP/TCP vs QUIC protocol.

For further analysis, a comparison will be made to the implementation of the congestion control algorithm in the QUIC protocol to see the performance impact on execution time. Based on Figure 6, there are contrasting patterns for the New Reno and Vegas algorithms compared to the BIC and QUIC BBR algorithms. For the new Reno and Vegas algorithms, the execution time reaches 6.25221 seconds for

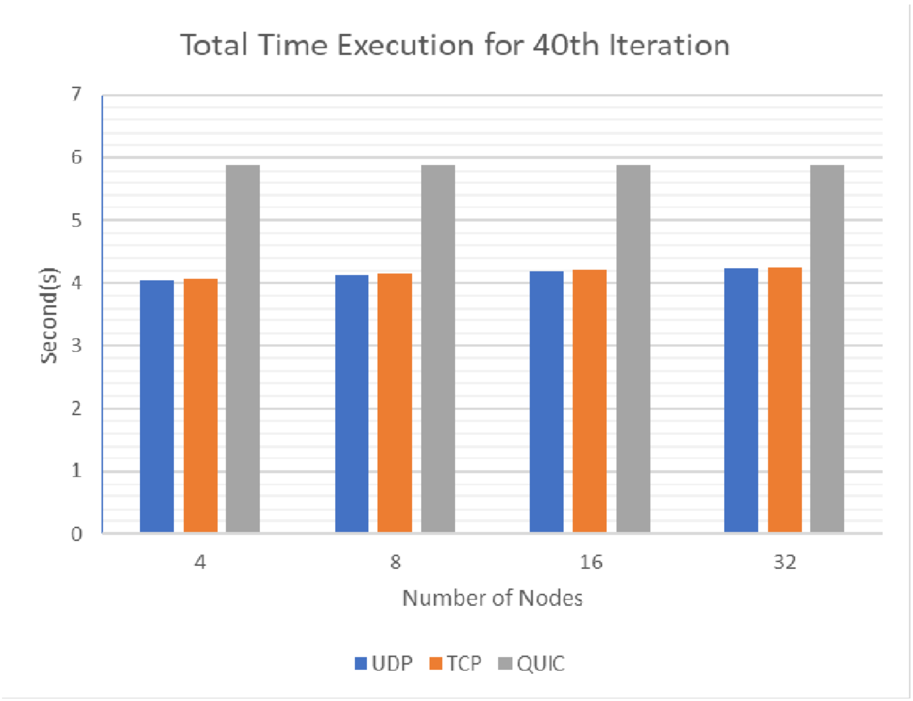

Figure 5. Average Iteration Time for TCP vs QUIC Protocol

40 iterations, with a difference of almost 0.5 seconds against the BIC and QUIC BBR algorithms with an execution time of 5.89087 seconds. The QUIC BBR algorithm is congestion control specifically for the QUIC protocol while the TCP BIC algorithm used is a TCP congestion module that is compatible with the implementation of the QUIC protocol.

6. Conclusions

Protocol is the foundation technology in ICT communication technology that allows communication between two nodes. We can ensure that data transport is successful using this protocol technology. The advancement of protocol technology can compensate for prior technological shortcomings and improve data transmission.

The PBFT algorithm excessively exchanges messages between nodes to preserve network integrity, the ability to transfer UDP datagrams with the congestion control capabilities of TCP becomes vital component. Furthermore, QUIC's ability to encrypt UDP datagrams ensures the security of communications within the blockchain ecosystem.

In this paper, comparison between UDP, TCP and QUIC protocol in blockchain infrastructure in terms of time needed for established consensus in PBFT algorithm. Based on the simulation that has been conduct in Section III we found that in our set number of nodes UDP and TCP showing better time execution than QUIC Protocol,

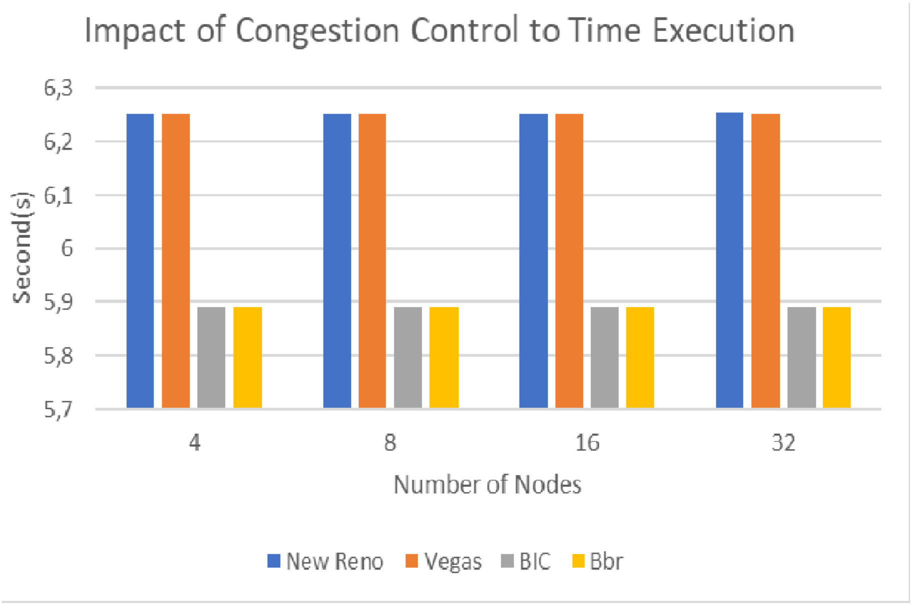

Figure 6. Impact of Congestion Control to Time Execution

however QUIC showing that addition number of nodes not giving significant time execution compared to TCP Protocol, we assume that in larger set number of nodes there are turning point that QUIC Protocol have better time execution than TCP Protocol. Furthermore, QUIC Protocol still have plenty of room for improvement that proved by changing congestion algorithms give result better time execution. In our study result, QUIC BBR and TCP BIC give better improvement.

To show more in-depth performance of QUIC Protocol in PBFT consensus, we deducted that need deeper investigation in existing communication algorithm to leveraging QUIC protocol as transport layer and testing with more nodes to prove that turning point QUIC Protocol vs TCP Protocol in larger number of dataset.

References

[1] Michele Polese et al. “A Survey on Recent Advances in Transport Layer Protocols”. In: *Commun. Surveys Tuts.* 21.4 (Oct. 2019), pp. 3584–3608. ISSN: 1553-877X. DOI: 10.1109/COMST.2019.2932905. URL: https://doi.org/10.1109/COMST.2019.2932905.

[2] Péter Megyesi, Zsolt Krämer, and Sándor Molnár. “How quick is QUIC?” In: *2016 IEEE International Conference on Communications (ICC)*. 2016, pp. 1–6. DOI: 10.1109/ICC.2016.7510788.

[3] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. In: *Decentralized business review* (2008).

[4] Peter Foytik et al. “A Blockchain Simulator for Evaluating Consensus Algorithms in Diverse Networking Environments”. In: *2020 Spring Simulation Conference (SpringSim)*. 2020, pp. 1–12. DOI: 10.22360/SpringSim.2020.CSE.003.

[5] Saif Talib Albasrawi. “Performance analysis of Google’s Quick UDP Internet Connection Protocol under Software Simulator”. In: *Journal of Physics: Conference Series* 1591.1 (July 2020), p. 012026. DOI: 10.1088/1742-6596/1591/1/012026. URL: https://dx.doi.org/10.1088/1742-6596/1591/1/012026.

[6] Alvise De Biasio et al. “A QUIC Implementation for Ns-3”. In: *Proceedings of the 2019 Workshop on Ns-3*. WNS3 ’19. Florence, Italy: Association for Computing Machinery, 2019, pp. 1–8. ISBN: 9781450371407. DOI: 10.1145/3321349.3321351. URL: https://doi.org/10.1145/3321349.3321351.

[7] Miguel Castro, Barbara Liskov, et al. “Practical byzantine fault tolerance”. In: *OsDI*. Vol. 99. 1999. 1999, pp. 173–186.

[8] Yaroslav Meshcheryakov et al. “On Performance of PBFT Blockchain Consensus Algorithm for IoT-Applications With Constrained Devices”. In: *IEEE Access* 9 (2021), pp. 80559–80570. DOI: 10.1109/ACCESS.2021.3085405.

[9] Shijie Zhang and Jong-Hyouk Lee. “Analysis of the main consensus protocols of blockchain”. In: *ICT Express* 6.2 (2020), pp. 93–97. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2019.08.001. URL: https://www.sciencedirect.com/science/article/pii/S240595951930164X.

[10] URL: https://www.nsnam.org/docs.

[11] Rishav Halder et al. “NS3TCG: NS3 Topology and Code Generator”. In: *2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)*. 2018, pp. 865–870. DOI: 10.1109/ICRIEECE44171.2018.9008653.

[12] Zhayujie. *Blockchain Simulator*. https://https://github.com/zhayujie/blockchain-simulator. 2022.

[13] Wei Bi, Huawei Yang, and Maolin Zheng. “An accelerated method for message propagation in blockchain networks”. In: *arXiv preprint arXiv:1809.00455* (2018).

[14] Du Mingxiao et al. “A review on consensus algorithm of blockchain”. In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2017, pp. 2567–2572. DOI: 10.1109/SMC.2017.8123011.

[15] Wenyu Li et al. “A Scalable Multi-Layer PBFT Consensus for Blockchain”. In: *IEEE Transactions on Parallel and Distributed Systems* 32.5 (2021), pp. 1146–1160. DOI: 10.1109/TPDS.2020.3042392.