DBPF: An Efficient Dynamic Block Propagation Framework for Blockchain Networks

Osama Farouk[1], Mahmoud Bakrey[2], Mohamed Abdallah[3]
Department of Information System-Faculty of Computers and Informatics, Suez Canal University, Ismailia, Egypt, 41522[1,2]
School of Computing and Data Science, BADYA University, Cairo, Egypt[3]

Abstract*—Scalability poses a significant challenge in blockchain networks, particularly in optimizing the propagation time of new blocks. This paper introduces an approach, termed "DBPF" - Dynamic Block Propagation Framework for Blockchain Networks, aimed at addressing this challenge. The approach focuses on optimizing neighbor selection during block propagation to mitigate redundancy and enhance network efficiency. By employing informed neighbor selection and leveraging the Brotli lossless compression algorithm to reduce block size, the objective is to optimize network bandwidth and minimize transmission time. The DBPF framework calculates the Minimum Spanning Tree (MST) to ensure efficient communication paths between nodes, while the Brotli compression algorithm reduces the block size to optimize network bandwidth. The core objective of DBPF is to streamline the propagation process by selecting optimal neighbors and eliminating unnecessary data redundancy. Through experimentation and simulation of the block propagation process using(DBPF), we demonstrate a significant reduction in the propagation time of new blocks compared to traditional methods. Comparisons against approaches such as selecting neighbors with the least Round-Trip Time *RTT*, random neighbor selection, and the DONS approach reveal a notable decrease in propagation time up to more than (45%) compared to them based on network type and number of nodes. The effectiveness of (DBPF) in boosting blockchain network efficiency and decreasing propagation time is emphasized by the experimental findings. Additionally, various compression algorithms such as zstandard and zlib were tested during the research. Nevertheless, the results suggest that Brotli produced the most positive outcomes. Through the integration of optimized neighbor selection and effective data compression, DBPF presents a hopeful resolution to the scalability issues confronting blockchain networks. These results showcase the capability of (DBPF) to notably enhance network performance, leading the path toward smoother and more efficient blockchain operations

Keywords*—*Blockchain; scalability; minimum spanning tree; compression; broadcasting; optimized neighbor selection; network bandwidth; transmission time optimization

I. Introduction

Blockchain (BC) technology has revolutionized the way data and financial transactions are shared among network participants BC was initially introduced in 2008, credited to Satoshi Nakamoto [1]. At its core, a blockchain is a decentralized and distributed ledger that securely records transactions across a network of nodes. The value added by blockchain technology lies in its ability to provide transparency, immutability, and trust. By eliminating the need for intermediaries and central authorities, blockchain enables peer-to-peer transactions and data sharing, fostering a new era of decentralized applications and services [2], [3]. This improves its reliability and efficiency compared to traditional data storage systems. The networks within (BC) can securely manage information and protect it from tampering, even in the presence of some malicious nodes, These features are highly valuable and find applications not only in cryptocurrencies but also across a wide range of fields [4]. Therefore, (BC) has many applications in emerging fields such as the internet of things [5], [6], 5G [7] [8], [9], and artificial intelligence, [10], [11].

However, despite its numerous advantages, blockchain faces scalability challenges that hinder its widespread adoption [12]. One of the key scalability issues is the low throughput of blockchain networks, primarily caused by the high propagation time required to disseminate new blocks to all nodes in the network. This delay in block propagation can lead to inconsistencies in the blockchain, impacting transaction finality and network performance. It is evident that Bitcoin's transaction processing capability is limited, allowing for only seven Transactions Per Second (TPS). This stands in stark contrast to widely adopted mainstream payment platforms like PayPal, which achieves a transfer rate of 500 TPS, and Visa, surpassing 4000 TPS [13]. Similarly, **Ethereum**, another prominent cryptocurrency, can handle approximately 15 TPS. Clearly, both **Bitcoin** and **Ethereum** are inadequate when it comes to meeting the demands of large-scale trading scenarios.

The high propagation time in blockchain networks can be attributed to several factors. Firstly, traditional methods used for path selection and neighbor discovery in blockchain networks often result in suboptimal routing paths, leading to longer propagation times and message redundancy. Random Neighbor Selection (RNS) is one such method that may contribute to inefficient block dissemination by selecting neighbors without considering the network topology or optimal paths, where shared data propagates through random paths [14], leading to an inefficient data propagation scheme. This inefficiency arises from the probability of redundancy in the exchanged messages between network nodes. This redundancy arises from cycling in randomly chosen data paths, leading to longer delivery times and reduced consistency. Despite this, most blockchain (BC) systems support Random Neighbor Selection (RNS). Several methods have been proposed to enhance the Neighbor Selection (NS) process locally, addressing the issue of dynamicity. For example, Bi et al. [15] introduced an NS protocol based on network latency, where nodes measure the Round Trip Time (RTT) to their neighboring nodes and prioritize those with the lowest RTT for the NS process. However, none of these methods has presented an optimal NS strategy.

Additionally, the size of blocks and network bandwidth limitations also play a significant role in increasing propagation time. As block sizes grow, the time required to propagate blocks across the network increases proportionally [16]. Moreover, some blockchain protocols overlook the optimization of network bandwidth usage, further exacerbating propagation delays.

In response to these challenges, this paper proposes a novel solution to enhance blockchain scalability by addressing the inefficiencies in neighbor selection and block size management. Our approach focuses on optimizing neighbor selection for each node in the network (ONS) by calculating the Minimum Spanning Tree (MST) to establish efficient communication paths. Furthermore, we aim to optimize network bandwidth by reducing block size and improving propagation speed by implementing the (Brotli compression) method, a lossless data compression technique that minimizes redundant data within blocks and transactions. By combining optimized neighbor selection and block size reduction through compression, our proposed solution (**DBPF**) seeks to mitigate the propagation time challenges in blockchain networks, ultimately enhancing network scalability and performance.

The contributions of this paper are summarized as follows:

1) The Dynamic Block Propagation Framework (DBPF) addresses blockchain scalability by optimizing neighbor selection through a Minimum Spanning Tree (MST), reducing redundant data propagation, and improving communication efficiency between nodes.
2) DBPF incorporates Brotli lossless compression to reduce the size of the propagated blocks, improving network bandwidth usage and speeding up block dissemination.
3) By leveraging MST-based Optimal Neighbor Selection, DBPF selects neighbors that enable faster and more efficient block propagation, reducing the overall time needed to disseminate blocks across the network.
4) The framework significantly reduces the total propagation time for data exchanged between nodes, ensuring faster and more efficient block distribution.
5) DBPF combines optimized neighbor selection with effective data compression, providing a scalable and efficient solution for large blockchain systems to increase the throughput of the network.

The remaining sections of this paper are structured as follows: Section II analyzes relevant literature, Section III provides the proposed system model for DBPF, Section IV provides a detailed explanation of the proposed DBPF, Section V presents the evaluation of DBPF, and finally, Section VI summarizes the most significant findings and conclusions.

II. Related Work

This section presents several modern network layer scalability solutions, which primarily aim to enhance either the gossip algorithm or reduce block data size. Research studies targeting improvements in the gossip algorithm focus on reducing duplicate data and increasing block propagation speed [17]. These proposed solutions aim to minimize duplication caused by the gossip protocol decrease block propagation time through enhanced gossip mechanisms and reduce the size of the block to accelerate block propagation through network bandwidth. Below are some recent works that represent such solutions.

The authors of “PiChu: Accelerating Block Broadcasting in Blockchain Networks with Pipelining and Chunking” [18] introduce an innovative method to enhance the efficiency and scalability of blockchain networks. PiChu leverages pipelining and chunking techniques to expedite block propagation, significantly reducing the time required to broadcast blocks across the network. The approach involves verifying the block header before dividing the block into smaller segments, which are then processed and forwarded in parallel. This continuous data flow minimizes delays associated with waiting for entire blocks to be verified. The chunking process splits each block into smaller, self-contained chunks that include complete transactions, allowing for incremental verification and faster dissemination. PiChu is particularly effective in scenarios with high transaction volumes and large block sizes, making blockchain networks more robust and scalable. However, PiChu’s effectiveness is influenced by the network topology and the connectivity between nodes. The requirement for modifications to existing consensus protocols may also limit its immediate applicability in some blockchain systems. Additionally, the benefits of pipelining and chunking may vary depending on the specific characteristics of the network, and The need for nodes to verify and forward chunks can introduce additional processing overhead.

Baniata and Anaqreh [19] introduced the Dynamic Optimized Neighbor Selection Algorithm (DONS) to enhance P2P network management within blockchain networks. In this approach, a leader peer is selected to manage the network and construct its topology using neighbor lists from regular peers. The leader uses a Minimum Spanning Tree (MST) to identify optimal neighbors, thereby minimizing propagation delay and enhancing transaction throughput. However, when the leader changes, the network topology must be reconstructed, necessitating the collection of neighbor lists anew. As the number of peers increases, the time required to compute the MST also increases, leading to inefficient bandwidth utilization. Furthermore, the unavailability of the leader poses risks of topology loss and incurs overheads associated with leader reselection.

Zhang et al. [20] investigate the dynamics of block propagation in blockchain-based vehicular networks (VANETs) and the impact of node mobility on blockchain consensus mechanisms. The authors propose an analytical model to derive a closed-form expression for single-block propagation time and analyze multi-block competitive propagation to address blockchain forking issues. The model accounts for the dynamic connectivity of moving nodes, introducing opportunistic communication to blockchain consensus. Their findings reveal that higher mobility and more moving vehicles expedite block propagation, thereby reducing consensus time. The study also shows that distinct propagation capabilities of moving nodes help mitigate blockchain forking. However, the model has limitations, such as its reliance on closed-form expressions and the assumption of a single-chain structure. The approach may face limitations in scenarios with highly dynamic topologies and varying node densities, which could affect its overall performance.

The paper “FastChain: Scaling Blockchain System with

Informed Neighbor Selection" by Ke Wang and Hyong S. Kim [21] introduces the FastChain protocol to enhance blockchain scalability by optimizing block propagation times. FastChain operates by leveraging nodes with higher bandwidth capacity to distribute blocks throughout the network. Nodes with limited bandwidth prioritize connections with high-bandwidth nodes and disconnect from those with bandwidth below a specific threshold. This mechanism comprises two essential stages: the bandwidth monitoring phase and the neighbor update phase. During the bandwidth monitoring phase, each node maintains a table containing the recent bandwidth information of its neighboring nodes. In the neighbor update phase, nodes periodically update their connections, disconnecting from those with low bandwidth. This informed neighbor selection policy significantly reduces block propagation time, thereby increasing the effective block rate and improving throughput by 20% to 40%. Despite its advantages, the FastChain protocol introduces additional processing overhead due to the continuous monitoring and updating of neighbor connections. Moreover, reliance on high-bandwidth nodes could lead to centralization risks within the network.

The authors in [22], [23] proposed a score-based neighbor selection protocol for constructing a blockchain network. This protocol assigns higher scores to peers with lower propagation delays compared to those with higher propagation delays. Subsequently, peers with the highest scores are chosen as neighbors. Each miner node evaluates its neighboring nodes based on the time difference between when the block was created and when it was received by the recipient node. After successfully receiving ten blocks, a node updates its list of neighbors. During this update, the node randomly selects new neighbors, including only those with high scores. Neighbor nodes that exhibit faster block transfer rates compared to others are assigned higher scores, indicating superior network communication capabilities. Thus, miners prefer these high-scoring neighbors. This method leads to excessive dependence on the nodes that have the shortest total propagation time, which can reduce node performance.

Wang introduces the Txilm protocol in [24] to tackle the challenge of large data transmissions in blockchain networks. This is achieved through the application of lossy block compression alongside salted short hashing. The protocol operates by pre-sorting transactions based on their identifiers (TXIDs) or other criteria and hashing them using a short hash function combined with a cryptographic salt. This process significantly reduces the data size, resulting in substantial bandwidth savings. The compressed transaction list is broadcast instead of the original transactions, achieving up to 100x bandwidth efficiency. In cases of hash collisions, a second-stage resolution involving Merkle tree recomputation ensures data integrity. While the protocol effectively reduces data transmission sizes and enhances network scalability, it introduces additional computational overhead for collision resolution and depends on the consistent ordering of transactions and mempool size across nodes.

III. The Proposed System Model

To address the scalability and efficiency challenges in blockchain networks, we propose the Dynamic Block Propagation Framework (DBPF). This framework integrates MST-based optimal neighbor selection with Brotli compression for block data, aiming to minimize propagation time and enhance network throughput.

Let $G = (V, E)$ represent the blockchain network, where V is the set of nodes and E is the set of edges. Each edge $(i, j) \in E$ is associated with a weight w_{ij}, indicating the latency or communication cost between nodes i and j. The MST $T = (V, E_T)$ is a subset of E that spans all nodes V with the minimum total edge weight. The MST is computed using Kruskal's algorithm, formulated as:

$$\min \sum_{(i,j)\in E_T} w_{ij}$$

The MST provides the optimal neighbor selection (ONS_i) for each node i.

Let B denote the block containing T transactions. The block size is given by:

$$\|B\| = \sum_{i=1}^{T} \|t_i\|$$

Applying Brotli compression reduces the block size to $C(B)$, with the compression ratio $R = \frac{\|B\|}{C(B)}$. The propagation time P_t for a block B is determined by the compressed block size $C(B)$ and the network bandwidth W:

$$P_t = \frac{C(B)}{W}$$

For the entire network, the total propagation delay Δt considering the MST is given by:

$$\Delta t = \sum_{(i,j)\in E_T} \frac{C(B)}{W_{ij}}$$

where W_{ij} is the bandwidth between nodes i and j.

The overall objective of DBPF is to minimize the total propagation time by integrating MST-based optimal neighbor selection and Brotli compression:

$$\min_{T,C(B)} \Delta t = \min_{T,C(B)} \sum_{(i,j)\in E_T} \frac{C(B)}{W_{ij}}$$

This model demonstrates how DBPF optimizes both the network topology and data propagation processes, thereby enhancing blockchain network efficiency and scalability.

IV. The Proposed Efficient Dynamic Block Propagation Framework (DBPF)

This section introduces the explanation and design of an Efficient Dynamic Block Propagation Framework for Blockchain Networks (DBPF), which incorporates a combination of an efficient algorithm in neighbor selection by obtaining optimal neighbor for each node in the network by constructing a minimum spanning tree (MST) and block size management by using **Brotli** compression algorithm to optimize network bandwidth. The primary objective of the DBPF protocol is to reduce message propagation time, optimize network bandwidth utilization, elevate overall network performance by leveraging compression techniques on block messages, and optimize blockchain network construction to be a solution for blockchain scalability problems.

To provide a clear understanding of the proposed solution, a diagram in Fig. 1 is presented. This diagram highlights the main components and mechanisms of the solution (DBPF), focusing on optimized neighbor selection and block size reduction through compression.

A. Phase-1: Leader Selection

In the context of the "**DBPF**", the leader Selection process adopts a random selection approach to designate a node as the leader within the blockchain network. This phase encompasses two primary tasks: Network leader selection and leader announcement. The DBPF requires a global view of the BC network. All nodes in BC have equal privileges in the public and permissionless BC network. However, the proposed **DBPF** selects one of these nodes to be the leader node (**LN**) to perform **MST** calculations for all other nodes. **LN** have advantages compared to other nodes in the same (BC) network, allowing it a global view or metadata information of the entire network. Additionally, **LN** collects information from the other nodes within the same network and uses it to generate the **MST** for the entire Network. Thus, each node in the network can select its optimal neighbors (ON) from the generated MST to exchange new blocks or transactions.

1) Step 1: Random selection: The **DBPF** protocol incorporates a random selection process for choosing a **LN**, which offers several notable advantages. Firstly, this approach ensures fairness by giving every node an equal opportunity to be selected as a leader, thereby eliminating potential biases. Secondly, the random selection method promotes unbiased network operations and decentralization, as it prevents the influence of network hierarchies or power dynamics in the leader selection process. This aligns with the core concept of decentralization within the BC ecosystem. Thirdly, in dynamic networks where nodes frequently join or leave, the ability to swiftly transition leadership is crucial. Random selection facilitates rapid leadership transitions without the delays associated with traditional election processes. If a **LN** departs or becomes less effective, a new leader can be randomly chosen, ensuring minimal disruption to network operations. This adaptability is paramount for maintaining the seamless functioning of the network in the face of changing circumstances.

In addition to the previously mentioned advantages, the random selection process in the **DBPF** protocol also enhances the security of the network. By randomly selecting a leader, the protocol introduces an element of unpredictability in the leadership position. Potential attackers or malicious actors are unable to predict or target specific nodes that might have elevated privileges as leaders. This randomness adds an extra layer of security to the network, as it prevents adversaries from exploiting any inherent vulnerabilities associated with predictable leader selection methods.

2) Step 2: Assessment of the Randomly Selected Leader: After a node is randomly selected as a potential leader, it undergoes a critical evaluation process to confirm its suitability. This assessment is vital to ensure that the node meets key criteria necessary for effective leadership. If the selected node meets these criteria, it is confirmed as the leader. If it does not, the random selection process is repeated. This step is designed to minimize the likelihood of choosing an unsuitable node, particularly aiming to avoid light nodes in favor of full nodes for leadership roles.

The first essential criterion is computational strength. The prospective leader node must possess substantial computational power to handle the responsibilities associated with leadership. This includes processing large volumes of data, making rapid and accurate decisions, and managing complex algorithms essential for optimizing the network. High computational capacity ensures that the node can perform its tasks efficiently without bottlenecks. Robust connectivity is another crucial requirement for the leader node. The selected leader must exhibit strong connectivity within the network. This robust connectivity is vital for ensuring efficient communication between the leader and other nodes, facilitating the swift and reliable dissemination of directives and information. Well-connected nodes enhance the overall responsiveness and coordination within the network. Finally, stability and reliability are essential characteristics of a leader node. The node must demonstrate a history of stable and reliable operation. High uptime and consistent availability are crucial, as they indicate the node's capability to maintain its leadership role without causing disruptions. A stable leader node ensures continuity and reliability in the network's functioning. To minimize the frequency of reselection, there is a preference for choosing from full nodes rather than light nodes. Full nodes, typically being more robust in terms of resources and connectivity, are more likely to meet the leadership criteria on the first selection. This approach streamlines the selection process, making it more efficient and reducing the time and resources spent in finding a suitable leader.

3) Step 3: Leadership Redetermination: In cases where the initially selected node does not meet the required criteria, the protocol stipulates a reselection. This process is promptly initiated to ensure that network leadership is established without undue delay.

Finally, The leader announcement proposed by the DBPF framework can be described as follows: Following the leader selection process, the DBPF notifies all nodes in the network about the new leader by sending announcement messages to all of them. Additionally, it informs the new leader of their responsibility for creating the MST for the network and broadcasting it to all nodes within the BC network. this phase algorithm is presented in Algorithm 1.

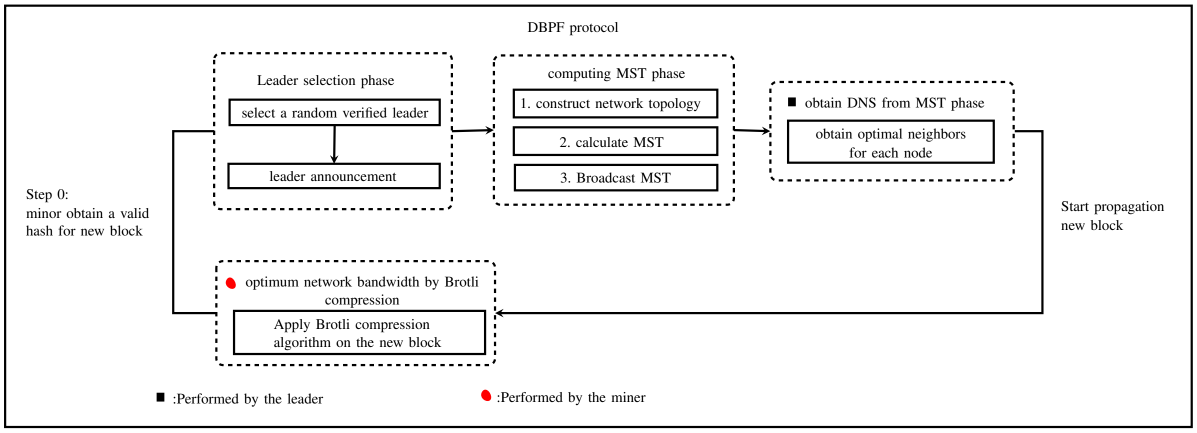

DBPF protocol
Leader selection phase
select a random verified leader
leader announcement
computing MST phase
1. construct network topology
2. calculate MST
3. Broadcast MST
■ obtain DNS from MST phase
obtain optimal neighbors for each node
Step 0: minor obtain a valid hash for new block
Start propagation new block
● optimum network bandwidth by Brotli compression
Apply Brotli compression algorithm on the new block
■ :Performed by the leader
● :Performed by the miner

Fig. 1. The main steps involved in the proposed DBPF framework.

Algorithm 1 Leader Selection in DBPF Protocol

```
Require: Blockchain Network $\mathcal{N}$
Ensure: Selected Leader Node $L$
 1: Begin
 2: RandomlySelectNode:
 3:   $n \leftarrow$ select a node randomly from $\mathcal{N}$
 4: if $\neg$ isFullNode($n$) then
 5:     goto step 2 (RandomlySelectNode)
 6: end if
 7: AssessCandidate:
 8: if hasComputationalStrength($n$) $\wedge$ hasRobustConnectivity($n$) $\wedge$ isStableAndReliable($n$) then
 9:     $L \leftarrow n$
10: else
11:     goto step 2 (RandomlySelectNode)
12: end if
13: End
```

B. Phase-2: Calculate and Dissemination of the Minimum Spanning Tree (MST)

In DBPF protocol, the subsequent phase following the random selection of the leader (L) involves a critical process that lays the foundation for network optimization. Upon phase 1, Each node and miner within the network acknowledges and recognizes the appointed leader (L). Consequently, Phase-2 is triggered and is performed by the leader (*L*) to compute MST for BC network (G) by using Kruskal's Algorithm as follows:

1) Step-1: Collection of Local information (LIs): L commences the process by collecting Local information (*LIs*) from all participating nodes (n) in the network (G). *LIs* consist of anonymized data regarding the network's topology, crucial for understanding the current network structure without compromising the privacy of individual nodes.

2) Step-2: Constructing Network Topology(NT): L employs the (*LIs*) stored locally to formulate an anonymized representation of the global network topology *NT*, Presented as an Adjacency List [25]. The Adjacency List is a fundamental data structure used to represent graph structures efficiently, particularly suitable for sparse graphs like BC networks, where the number of edges is significantly less than the number of possible edges. In the context of the Dynamic Block Propagation Framework (DBPF), the Adjacency Listis employed to construct the network topology (NT) based on the Local Information (LIs) collected from all participating nodes in the blockchain network. An Adjacency List represents a weighted graph $G = (V, E)$ where V is the set of vertices (nodes) and E is the set of weighted edges (connections with associated weights between nodes). Each vertex in the graph maintains a list of its adjacent vertices along with the weights of the connecting edges. This list captures all nodes directly connected to it by an edge and the corresponding weights, thus providing a compact and efficient way to store and manipulate the network topology.

Formally, for a weighted graph G with vertices $V = \{v_1, v_2, \ldots, v_n\}$ and weighted edges E, the Adjacency List is an array of lists, where the i-th element of the array corresponds to vertex v_i and contains a list of pairs (v_j, w_{ij}) such that there exists a weighted edge $(v_i, v_j) \in E$ with weight w_{ij}.

Example:

Consider a blockchain network with five nodes: $A, B, C, D,$ and E. The weighted connections (edges) between these nodes are as follows: $A - B(2), A - C(3), B - D(4), C - D(1), D - E(5)$. The Adjacency List representation of this weighted network would be:

- $A : \{(B, 2), (C, 3)\}$
- $B : \{(A, 2), (D, 4)\}$
- $C : \{(A, 3), (D, 1)\}$
- $D : \{(B, 4), (C, 1), (E, 5)\}$
- $E : \{(D, 5)\}$

This structure efficiently represents the network, allowing for quick look-up and traversal of adjacent nodes along with the weights of the connecting edges. Fig. 2 illustrates this weighted blockchain network graph.

The Adjacency List offers several advantages in the context of the DBPF. It provides an efficient way to store and access

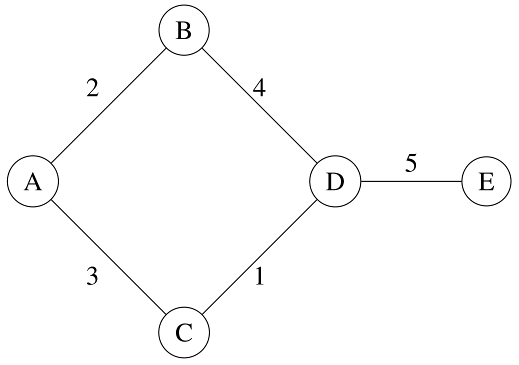


Fig. 2. Example of a weighted blockchain network graph.

the network structure, which is crucial for large blockchain networks with potentially thousands of nodes. As the blockchain network grows, the Adjacency List can handle the increasing number of nodes and edges without significant performance degradation, demonstrating its scalability. Furthermore, the list can be easily updated as new nodes join or leave the network which makes **DBPF** dynamic, ensuring that the representation remains accurate and up-to-date, highlighting its flexibility.

3) Step-3: Computation of the (MST) Using Kruskal's Algorithm:: Following the previous step where the leader node (L) utilized Local Information (LIs) to construct the network topology (NT), the next phase in the Dynamic Block Propagation Framework (DBPF) involves computing the (MST) for the blockchain network (G) using Kruskal's algorithm [26] as shown in Algorithm 2. Let $G = (V, E)$ be the graph where V is the set of vertices and E is the set of edges with weights $w : E \rightarrow \mathbb{R}$. The goal is to find a subset $T \subseteq E$ that forms a tree covering all vertices V with the minimum total edge weight:

$$\min \sum_{e \in T} w(e)$$

subject to T forming a connected acyclic subgraph of G.

Kruskal's algorithm sorts the edges E in non-decreasing order by weight and iteratively adds the shortest edge to T, provided it does not form a cycle, using the union-find data structure to manage disjoint sets. This structured approach ensures the MST is computed efficiently, forming a key component of our DBPF for optimizing blockchain network performance. The primary goal is to minimize the total edge weight while preserving network connectivity and avoiding cycles. Kruskal's algorithm processes each edge independently, which is advantageous when the number of edges $|E|$ is much smaller than the number of possible edges $\left(\frac{V(V-1)}{2}\right)$. This reduces the overall computational burden. The computed MST serves as the backbone for block propagation, significantly reducing propagation time and enhancing overall network efficiency. The time complexity of Kruskal's algorithm is $O(E \log E + E \log V)$, making it efficient for the large-scale and sparse nature of blockchain networks.

Algorithm 2 Compute MST using Kruskal's Algorithm

```
1: Input: Network topology $NT = (V, E)$
2: Output: Minimum Spanning Tree (MST)
3: procedure COMPUTE_MST(NT)
4:     Convert $NT$ to a dictionary of dictionaries
5:     Initialize $T \leftarrow \emptyset$    ▷ Initialize the set for MST edges
6:     Create a union-find data structure for $V$
7:     Extract nodes and edges from $NT$
8:     Sort the edges $E$ in non-decreasing order of their weights $w(e)$
9:     for each edge $e = (u, v, w)$ in sorted $E$ do
10:        if FIND(parent, u) $\neq$ FIND(parent, v) then
11:            $T \leftarrow T \cup \{e\}$    ▷ Add edge to MST
12:            UNION(parent, rank, u, v)
13:        end if
14:    end for
15:    return $T$    ▷ The set of edges in the MST
16: end procedure
```

Example

Consider a blockchain network with five nodes: A, B, C, D, E. The weighted connections (edges) between these nodes are as follows: $A-B(2)$, $A-C(3)$, $B-D(4)$, $C-D(1)$, $D-E(5)$, as shown in Fig. 3.

Edge	Weight
(A, B)	2
(A, C)	3
(B, D)	4
(C, D)	1
(D, E)	5

The steps to compute the MST are as follows:

1) **Initialization**: Nodes: {A, B, C, D, E}; Edges: {(A, B, 2), (A, C, 3), (B, D, 4), (C, D, 1), (D, E, 5)}
2) **Sorting**: Sorted Edges: {(C, D, 1), (A, B, 2), (A, C, 3), (B, D, 4), (D, E, 5)}
3) **Edge Selection**:
   - Add (C, D, 1): No cycle
   - Add (A, B, 2): No cycle
   - Add (A, C, 3): No cycle
   - Skip (B, D, 4): Forms a cycle
   - Add (D, E, 5): No cycle
4) **Result**: MST: {(C, D, 1), (A, B, 2), (A, C, 3), (D, E, 5)} as shown in Fig. 4.

4) Step-4: Optimal Neighbor Selection (ONS):: Following the computation of the Minimum Spanning Tree (MST) by the leader node (L) in the previous step, the subsequent step involves disseminating the MST to all nodes within the blockchain network. This step aims to enable each node to derive its Optimal Neighbor Selection (ONS) from the received MST. Upon receiving the MST, each node runs Algorithm 3 to identify its optimal neighbors. This algorithm examines the MST and selects the most efficient connections based on the weights of the edges. These optimal neighbors are then used by the nodes to efficiently transmit new blocks or transactions across the blockchain network.

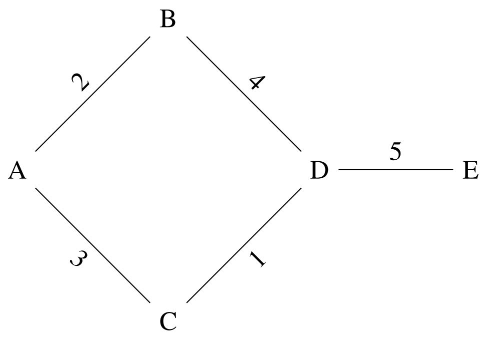


Fig. 3. Example of a weighted blockchain network graph.

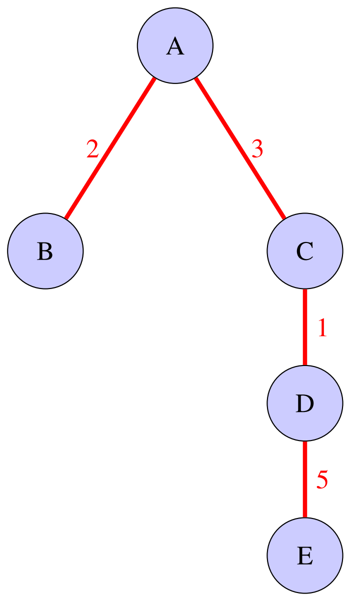


Fig. 4. Minimum spanning tree for the blockchain network.

This process ensures that data sharing is optimized, leveraging the MST to enhance overall network efficiency and performance. By utilizing the MST, the nodes can avoid redundant connections and minimize latency, which is crucial for maintaining the speed and reliability of blockchain operations. The MST-based ONS provides each node with a clear, efficient pathway for data transmission, contributing to the overall scalability and robustness of the blockchain network.

Algorithm 3 Find Optimal Neighbor Selection (ONS)

```
Require: MST $G = (V, E)$, node identifier node_id
Ensure: Optimal Neighbor Selection (ONS)
 1: ONS $\leftarrow \{\}$                          ▷ Initialize the set for ONS
 2: for each $u \in G$ do
 3:     if $u ==$ node_id then
 4:         for each neighbor $v_j \in G[u]$ do
 5:             ONS$[v_j] \leftarrow G[u][v_j]['weight']$
 6:         end for
 7:         break
 8:     end if
 9: end for
10: return ONS
```

C. Phase-3: Optimize Network Bandwidth by Brotli Compression

In this phase of the Dynamic Block Propagation Framework (DBPF), the focus shifts to optimizing network bandwidth and enhancing block propagation speed through the use of the Brotli lossless compression Algorithm [27]. This phase operates in parallel with the leader node's computation of the Minimum Spanning Tree (MST), enabling each node to transmit the compressed block efficiently to its Optimal Neighbors (ON). By doing so, the DBPF aims to reduce the propagation time of newly mined blocks across the network and increase the throughput of the public blockchain which contributes to solving the scalability problem.

The primary objective of compressing the block is to reduce its size ($\|B\|$), thereby minimizing the transmission time (P_t) across the network and optimizing the utilization of network bandwidth (W). This reduction in size aids in the faster propagation of the block to all nodes, addressing scalability issues and enhancing network efficiency.

We selected the Brotli compression algorithm for several reasons. Brotli, developed by Google, offers a high compression ratio and rapid decompression speed, making it particularly well-suited for The nature of blockchain data, which includes repetitive elements such as transaction inputs, outputs, and digital signatures, benefits greatly from Brotli's efficient compression techniques. A blockchain block is composed of two main parts: the header and the body. The header includes critical information necessary for the block's integrity and validation, such as the previous block hash, Merkle root, timestamp, difficulty target, and nonce. The body contains the transaction data, including inputs, outputs, and digital signatures. The block header's attributes are crucial for the validation cycle and maintaining the blockchain's integrity; thus, they are not compressed. Compressing these attributes would complicate validation and introduce security risks. Therefore, only the transaction data in the block body is compressed, preserving the essential information in the block header for seamless and secure validation. This compression process involves identifying redundant or less critical data within the block, applying the Brotli algorithm to reduce the block size, and then propagating the compressed block to neighboring nodes

The compression process involves several key steps. Initially, the miner identifies redundant or less critical data within the block, such as transaction inputs, outputs, digital signatures, and intermediate branches of the Merkle tree. Although the Merkle root must remain uncompressed for validation purposes, other components are suitable for compression. Let B represent a block containing T transactions, with the block size $\|B\|$ given by:

$$\|B\| = \sum_{i=1}^{T} \|t_i\|$$

Applying the Brotli compression algorithm, the block size is reduced to $C(B)$:

$$C(B) = \text{Brotli}(\|B\|)$$

The compression ratio R is defined as:

$$R = \frac{\|B\|}{C(B)}$$

This ratio reflects the effectiveness of the compression while ensuring that critical elements required for block validation, remain intact. Compressed transaction data is decompressed by nodes upon receipt, ensuring the block's integrity and validity are preserved.

When a miner successfully mines a new block B, it is compressed using the Brotli algorithm. This step occurs concurrently with the MST computation by the leader node. The miner then propagates the compressed block $C(B)$ through the optimized network to its neighboring nodes (N) which is calculated in phase 2. Each neighboring node decompresses $C(B)$ upon receipt to validate and add the block to its local blockchain. The propagation time (P_t) as a function of the compressed block size and network bandwidth is given by:

$$P_t = \frac{C(B)}{W}$$

By minimizing $C(B)$ through Brotli compression, the propagation time (P_t) is optimized, leading to enhanced network efficiency. Considering the network as a graph $G = (V, E)$, where V represents nodes and E represents communication links, the objective is to minimize the total propagation delay (Δt) across the network. This delay is influenced by the sum of individual propagation delays between nodes:

$$\Delta t = \sum_{i,j \in E} \frac{C(B)}{W_{ij}}$$

Here, W_{ij} represents the bandwidth between nodes i and j. By effectively compressing block data, DBPF optimizes the total propagation delay, improving the overall efficiency and scalability of the blockchain network.

By combining these two phases—optimized neighbor selection and enhanced network bandwidth—into a single framework called the Dynamic Block Propagation Framework (DBPF), we can significantly increase the throughput of blockchain networks. The DBPF leverages an MST-based optimal neighbor selection to establish efficient communication paths, alongside Brotli compression to reduce block sizes and transmission times. This dual approach not only minimizes propagation time but also enhances overall network efficiency. As we will demonstrate in the next evaluation section, DBPF outperforms many other methods in terms of scalability and performance in public blockchain networks.

V. Experiments and Results

This section provides a comprehensive overview of the experiments and assessments carried out to evaluate the Dynamic Block Propagation Framework (DBPF). It includes a discussion of the network datasets utilized, the performance metrics employed, and the detailed experiments executed. The network data was generated through a simulator that creates random network topologies using the Barabási-Albert(BA) model [28]. This model is well-suited for simulating real-world networks such as the Internet, social networks, and the World Wide Web.

To validate our analysis in real-world scenarios, we varied the network size and the average number of neighbors per miner. The simulation involves generating a random blockchain network where a miner node is randomly chosen as the source node for a data block. This source node disseminates the block to its neighboring nodes, which in turn propagate it to their neighbors, creating a cascading dissemination effect. The simulation ends once the block has reached all nodes in the network.

The experiments were run on a DELL PC equipped with an Intel(R) Core(TM) i5-10210U CPU (8 cores, 2.11 GHz), 8 GB DDR4-SDRAM, and a 500 GB SSD, running Windows 10 OS. We evaluated the experimental results using the Total Propagation Time (TP) metric in microseconds (μs), which measures the time it takes for a block sent from a randomly selected miner node to reach all nodes in the network.

Our experiments aim to demonstrate that the DBPF solution significantly enhances network performance by reducing the propagation time of new blocks, thereby improving the scalability and efficiency of blockchain networks.

The experiments are categorized as follows:

Experiment 1:

This experiment compares the performance of DBPF in terms of Total Propagation Time (TP) and the number of exchanged blocks (NB), against other algorithms such as Dynamic Optimized Neighbor Selection (DONS), Round-Trip Time (RTT), and Random Neighbor Selection (RNS). DONS is designed to optimize the propagation delay by selecting the most efficient paths based on delay metrics. RTT selects neighbors based on the minimum round-trip time, aiming to reduce latency in block propagation. RNS, on the other hand, randomly selects neighbors without considering network topology or path efficiency.The experiments were conducted using a network simulator based on the Barabási-Albert (BA) model to generate random network topologies. Two configurations were tested: networks with an average of 7 neighbors per node and networks with an average of 15 neighbors per node. The network bandwidth was assumed to be 10,000 bits per second. In each experiment, a randomly selected miner propagated a new block, and the propagation time and number of exchanged blocks were recorded.

The results demonstrate that DBPF significantly outperforms DONS, RTT, and RNS in both configurations. For networks with seven neighbors per node, DBPF achieved a total propagation time of 105.89 ms for 50 nodes, compared to 268.89 ms for DONS, 2124.58 ms for RTT, and 7489.9 ms for RNS. This trend continued as the network size increased to 100 and 450 nodes, with DBPF maintaining the lowest propagation times. Similarly, for networks with 15 neighbors per node, DBPF's propagation time remained consistently lower than the other algorithms. For example, with 450 nodes, DBPF achieved 156.95 ms compared to 840.50 ms for DONS, 19918.79 ms

TABLE I. Performance of DBPF Against DONS, RTT, and RNS

Nodes	Avg. Neighbors	Total Propagation Time (ms)			
		DBPF	DONS	RTT	RNS
50	7	105.89	268.89	2124.58	7489.9
100	7	263.37	721.72	5790.9	21089.09
450	7	326.42	1089.54	34805.4	162987.06
50	15	109.97	301.84	2152.55	14527.94
100	15	94.97	361.81	3271.19	21095.96
450	15	156.95	840.50	19918.79	125417.10

for RTT, and 125417.10 ms for RNS. The minimal increase in propagation time with DBPF highlights its efficiency and scalability. Additionally, DBPF maintained the lowest number of exchanged blocks as there is no any redundant block sent within the propagation process, indicating superior bandwidth utilization and network performance.

Based on the results in Table I, the insights derived from the experimental results underscore the superior performance of the Dynamic Block Propagation Framework (DBPF) in terms of propagation time and bandwidth efficiency compared to DONS, RTT, and RNS. As the number of nodes increased from 50 to 450, the propagation time for DBPF showed a minimal increase, highlighting its scalability and efficiency. For instance, DBPF's propagation time increased by only 220.53 ms (from 105.89 ms to 326.42 ms) for networks with seven neighbors per node, whereas RTT's time surged by 32680.82 ms (from 2124.58 ms to 34805.4 ms), and RNS's time skyrocketed by 155497.16 ms (from 7489.9 ms to 162987.06 ms). This demonstrates DBPF's superior scalability and efficiency in larger networks.

Compared to DONS, DBPF also showed significant improvements. For networks with seven neighbors per node, DBPF reduced the propagation time by 60.61% for 50 nodes, 63.52% for 100 nodes, and 70.05% for 450 nodes. Similar trends were observed for networks with 15 neighbors per node. This significant reduction in propagation time showcases the effectiveness of DBPF in optimizing network performance and achieving fast propagation of new blocks to all nodes.

The performance of the proposed DBPF algorithm was compared with DONS, RTT, and RNS algorithms across two experimental setups. The total propagation time for each algorithm as the number of nodes increases is shown in Fig. 5.

In Experiment 1 (Fig. 5a), with an average of seven neighbors per node, DBPF significantly outperforms the other algorithms. For instance, with 450 nodes, DBPF achieves a propagation time of 326.42 ms, whereas DONS, RTT, and RNS require 1089.54 ms, 34805.4 ms, and 162987.06 ms, respectively.

In Experiment 2 (Fig. 5b), with an average of 15 neighbors per node, DBPF again demonstrates superior performance. With 450 nodes, DBPF has a propagation time of 156.95 ms, compared to DONS at 840.50 ms, RTT at 19918.79 ms, and RNS at 125417.10 ms.

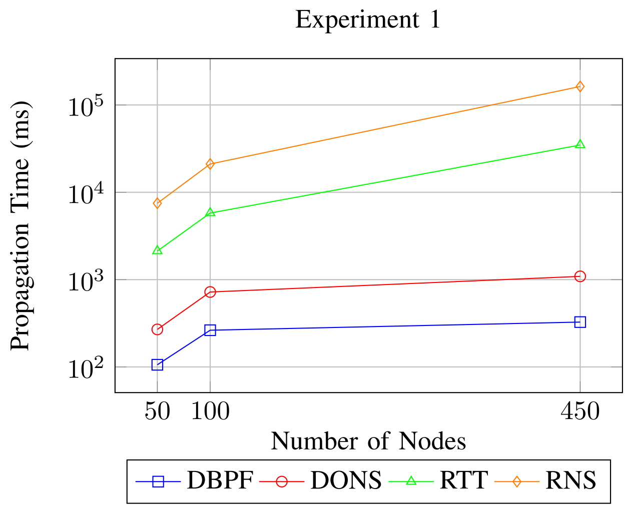
Experiment 2:

This experiment aims to demonstrate the robustness and efficiency of the Dynamic Block Propagation Framework (DBPF) under varying network bandwidth conditions, simulating realistic blockchain network scenarios under the Barabási-Albert (BA) model. Unlike previous experiments with constant bandwidth, this experiment introduces random bandwidth variations between nodes to assess DBPF's performance in more dynamic environments. The network configurations for this experiment include two cycles: Cycle 1 with an average number of neighbors per node set to 8 and Cycle 2 with an average number of neighbors per node set to 15. For each cycle, three different network sizes are tested: 30, 70, and 180 nodes. For each configuration, a random network topology is generated, and random bandwidth values are assigned to each pair of nodes to mimic real-world conditions where network bandwidth can vary. The total propagation time (PT) and the number of exchanged blocks (NB) are measured for DBPF, DONS, RTT, and RNS algorithms. The primary objective is to verify DBPF's efficiency in environments with fluctuating bandwidth and to compare its performance against other well-known algorithms (DONS, RTT, RNS). By simulating realistic network conditions, we aim to highlight DBPF's adaptability and robustness in maintaining low propagation times and efficient block propagation, even when network bandwidth varies significantly. The results from these experiments are presented in Table II.

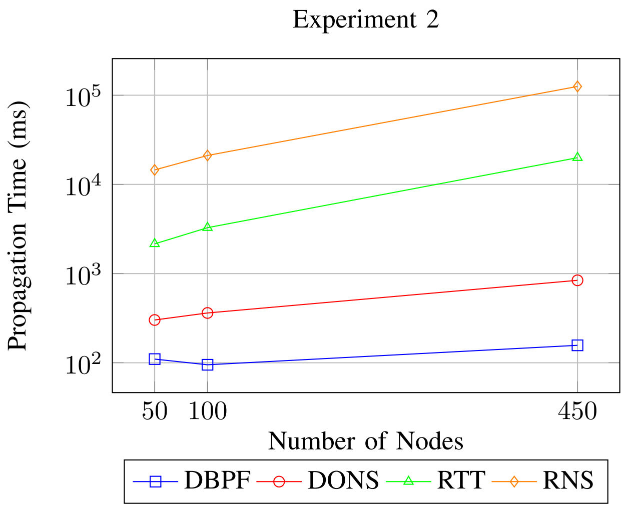
The results of the second experiment, conducted under varied bandwidth conditions, provide a comprehensive evaluation of the performance of the Dynamic Block Propagation Framework (DBPF) compared to DONS, RTT, and RNS. The experiments were conducted on two different average neighbor settings (8 and 15) with randomly generated bandwidth values to simulate realistic network conditions. The bandwidth values for the first set of experiments ranged from 1000 to 10000 Bps, while the second set ranged from 5000 to 100000 Bps.

The experiments clearly demonstrate the impact of bandwidth variability on propagation time across different algorithms. In scenarios with lower average neighbors (8), DBPF consistently outperformed other methods. For instance, with 70 nodes, DBPF achieved a propagation time of 140.46 ms compared to DONS (647.99 ms), RTT (5406.97 ms), and RNS (16553.99 ms). This indicates that DBPF is highly efficient in utilizing available bandwidth, leading to reduced propagation times.

In scenarios with higher average neighbors (15), DBPF continued to show superior performance. With 30 nodes, DBPF recorded a propagation time of 93.91 ms, significantly lower than DONS (154.92 ms), RTT (1121.16 ms), and RNS (4998.62 ms). This trend persisted with 70 nodes and 180 nodes, where DBPF consistently demonstrated lower propagation times, highlighting its robustness in handling varying network conditions and higher node densities.



(a) Avg. Neighbors = 7



(b) Avg. Neighbors = 15

Fig. 5. Comparison of total propagation time for different algorithms in two experimental setups

TABLE II. Performance of DBPF, DONS, RTT, and RNS under Varied Bandwidth Conditions (BA Model)

Model	Nodes	Avg. Neighbors	Total Propagation Time (ms)			
			DBPF	DONS	RTT	RNS
BA	30	8	160.26	630.43	1609.75	10345.17
BA	70	8	140.46	647.99	5406.97	16553.99
BA	180	8	219.80	865.98	18618.33	63783.90
BA	30	15	93.91	154.92	1121.16	4998.62
BA	70	15	111.12	167.33	2084.82	10820.95
BA	180	15	130.91	217.12	5694.26	44358.25

To quantitatively assess the performance improvement of DBPF, we calculate the percentage reduction in propagation time compared to DONS. For instance, with 180 nodes and 8 average neighbors, DBPF achieved a propagation time of 219.80 ms, while DONS recorded 865.98 ms. The percentage improvement is calculated as follows:

$$\text{Percentage Improvement} = \left(\frac{\text{DONS Time} - \text{DBPF Time}}{\text{DONS Time}} \right) \times 100$$

$$\text{Percentage Improvement} = \left(\frac{865.98 - 219.80}{865.98} \right) \times 100 \approx 74.61\%$$

Similarly, with 30 nodes and 15 average neighbors, DBPF showed a 39.38% improvement over DONS. These substantial improvements demonstrate the efficiency of DBPF in minimizing propagation time, and enhancing overall network performance.

Let T_{DBPF} and T_{DONS} represent the propagation times of DBPF and DONS, respectively. The percentage improvement $P_{\text{improvement}}$ can be expressed as:

$$P_{\text{improvement}} = \left(\frac{T_{\text{DONS}} - T_{\text{DBPF}}}{T_{\text{DONS}}} \right) \times 100$$

The experiments also showed significant differences in the number of exchanged blocks (NB) across different methods. For instance, DBPF required no redundant block exchanges compared to RTT and RNS, reflecting its efficiency in reducing network load and overhead

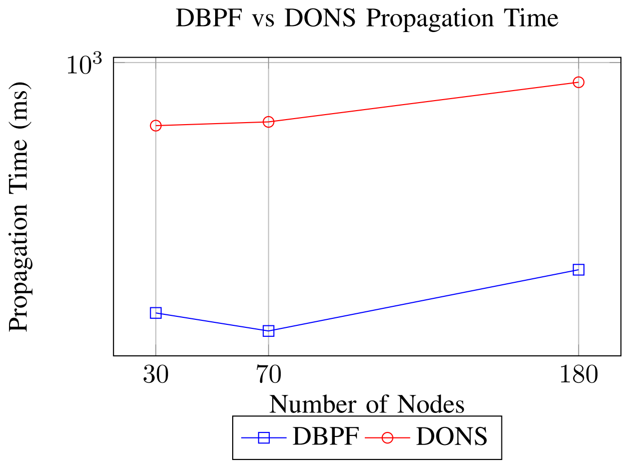


Fig. 6. Comparison of propagation time between DBPF and DONS.

These results underscore the effectiveness of DBPF in optimizing network performance, even under varying and challenging bandwidth conditions. The framework's ability to reduce propagation time significantly while maintaining low overhead highlights its potential for enhancing the scalability and efficiency of blockchain networks. As shown in Fig. 6, the DBPF algorithm consistently outperforms DONS across various network sizes and bandwidth conditions.

Experiment 3:

In this experiment, we investigate the efficiency of the Dynamic Block Propagation Framework (DBPF) implemented with different compression algorithms, specifically focusing on Brotli compression. The goal is to evaluate the performance of DBPF using Brotli in comparison with other widely-used compression techniques, zlib [29] and zstandard [30], under identical network conditions and the same block size. The Barabási-Albert (BA) model is employed to simulate realistic network environments.

The experiment consists of two cycles. In the first cycle, the number of nodes is set to 100 and 280, with an average number of neighbors per node fixed at 18. The bandwidth is kept constant at 10,000 bps to maintain a controlled environment. In the second cycle, the number of nodes is increased to 280 and 450, with the same average number of neighbors per node, but the bandwidth is randomly generated within the range of 1,000 to 10,000 bps to simulate dynamic network conditions.

The primary objective is to compare the performance of DBPF using Brotli, zlib, and zstandard compression algorithms. Zlib, a widely used compression library, provides a good balance between speed and compression ratio and is often utilized for data storage and transmission. Zstandard, developed by Facebook, offers high compression efficiency and fast decompression speeds, making it ideal for large-scale data compression tasks.

Metrics used to evaluate performance include total propagation time (ms). The hypothesis is that Brotli compression will outperform zlib and zstandard in reducing propagation time and optimizing network efficiency, demonstrating the robustness and scalability of the DBPF framework. The results from this experiment aim to provide a comprehensive analysis of the most efficient compression technique for enhancing blockchain network performance.

As it is shown in Table III, the results of this experiment highlight the effectiveness of the DBPF framework, particularly when using the Brotli compression algorithm. The propagation times for DBPF-Brotli were consistently lower than those for DONS and the other compression algorithms across all tested configurations. For instance, with 100 nodes and a fixed bandwidth of 10,000 bps, DBPF-Brotli achieved a propagation time of 105.65 ms, significantly outperforming DONS, which recorded 496.73 ms. Even when compared with other compression algorithms, DBPF-Brotli maintained its superiority, with DBPF-Zlib at 107.32 ms and DBPF-Zstandard at 216.80 ms. As the number of nodes increased to 280, DBPF-Brotli continued to show excellent performance, recording a propagation time of 91.45 ms, while DONS and DBPF-Zstandard recorded 627.63 ms and 297.06 ms, respectively. This trend was also observed when the bandwidth was randomly generated within the range of 1,000 to 10,000 bps. With 280 nodes, DBPF-Brotli achieved a propagation time of 163.32 ms, compared to DONS at 1496.8 ms, DBPF-Zlib at 171.27 ms, and DBPF-Zstandard at 189.71 ms. When examining the performance with 450 nodes under random bandwidth conditions, DBPF-Brotli again demonstrated superior performance with a propagation time of 99.65 ms. In contrast, DONS recorded 842.38 ms, DBPF-Zlib recorded 101.23 ms, and DBPF-Zstandard recorded 105.75 ms.

VI. Conclusion

The Dynamic Block Propagation Framework (DBPF) introduced in this study addresses key challenges in blockchain networks, such as limited transaction throughput, large blockchain sizes, scalability issues, and consensus protocol limitations. By integrating optimal neighbor selection (ONS) and advanced data compression techniques, specifically utilizing the Minimum Spanning Tree (MST) computation for efficient communication paths and the Brotli compression algorithm for data reduction, DBPF significantly enhances the performance of blockchain networks. Extensive experiments conducted using the Barabási-Albert (BA) model under various network conditions demonstrated the superiority of DBPF over existing methods. The outcomes of the presented research demonstrated a notable enhancement in block propagation across networks of diverse sizes, outperforming current state-of-the-art approaches. In networks with 180 nodes and 8 average neighbors, DBPF achieved up to a 74.61% improvement in propagation time compared to DONS. Further, DBPF outperformed RTT and RNS by 88.20% and 99.65%, respectively, under the same conditions. These results highlight the effectiveness of DBPF in reducing propagation time and enhancing overall network throughput. Additional experiments under varied bandwidth conditions confirmed the robustness and adaptability of DBPF. The framework maintained superior performance with both fixed and randomly generated bandwidths, demonstrating its flexibility in real-world scenarios. Evaluations with different compression algorithms, including Brotli, zlib, and zstandard, further underscored the efficiency of DBPF. Notably, DBPF using Brotli consistently outperformed other methods, validating the benefits of combining optimal neighbor selection with advanced compression techniques.

In summary, the DBPF framework offers a scalable and efficient solution for blockchain networks, significantly reducing block propagation time and increasing network throughput. These findings underscore the potential of DBPF to address inherent scalability challenges in blockchain technology, paving the way for more robust and efficient blockchain operations.

While the current study demonstrates the effectiveness of DBPF, several areas for future research can further enhance the framework's capabilities. Firstly, exploring the integration of DBPF with emerging consensus algorithms such as Proof of Stake (PoS) or Delegated Proof of Stake (DPoS) could provide insights into optimizing blockchain performance under different consensus mechanisms. future work could focus on implementing adaptive compression techniques that dynamically select the most suitable algorithm based on real-time network conditions and data characteristics.

References

[1] S. Nakamoto and A. Bitcoin, "A peer-to-peer electronic cash system," *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, vol. 4, no. 2, p. 15, 2008.

[2] O. Akanfe, D. Lawong, and H. R. Rao, "Blockchain technology and privacy regulation: Reviewing frictions and synthesizing opportunities," *International Journal of Information Management*, vol. 76, p. 102753, 2024.

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, 2014.

[4] J. Liu and J. Wu, "A comprehensive survey on blockchain technology and its applications," *Highlights in Science, Engineering and Technology*, vol. 85, pp. 128–138, 2024.

TABLE III. PERFORMANCE OF DBPF WITH DIFFERENT COMPRESSION ALGORITHMS

Nodes	Avg. Neighbors	Total Propagation Time (ms)			
		DBPF (Brotli)	**DONS**	**DBPF (Zlib)**	**DBPF (Zstandard)**
100	18	105.65	496.73	107.32	216.80
280	18	91.45	627.63	93.75	297.06
280	18 (Random BW)	163.32	1496.8	171.27	189.71
450	18 (Random BW)	99.65	842.38	101.23	105.75

[5] P. Danzi, A. E. Kalør, Č. Stefanović, and P. Popovski, "Delay and communication tradeoffs for blockchain systems with lightweight iot clients," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2354–2365, 2019.

[6] W. A. Al-Nbhany, A. T. Zahary, and A. A. Al-Shargabi, "Blockchain-iot healthcare applications and trends: a review," *IEEE Access*, 2024.

[7] I. Mistry, S. Tanwar, S. Tyagi, and N. Kumar, "Blockchain for 5g-enabled iot for industrial automation: A systematic review, solutions, and challenges," *Mechanical systems and signal processing*, vol. 135, p. 106382, 2020.

[8] D. Das, S. Banerjee, K. Dasgupta, P. Chatterjee, U. Ghosh, and U. Biswas, "Blockchain enabled sdn framework for security management in 5g applications," in *Proceedings of the 24th International Conference on Distributed Computing and Networking*, pp. 414–419, 2023.

[9] S. Onopa and Z. Kotulski, "State-of-the-art and new challenges in 5g networks with blockchain technology," *Electronics*, vol. 13, no. 5, p. 974, 2024.

[10] A. K. Tyagi and S. Tiwari, "The future of artificial intelligence in blockchain applications," in *Machine learning algorithms using scikit and tensorflow environments*, pp. 346–373, IGI Global, 2024.

[11] A. M. S. Saleh, "Blockchain for secure and decentralized artificial intelligence in cybersecurity: A comprehensive review," *Blockchain: Research and Applications*, p. 100193, 2024.

[12] A. I. Sanka and R. C. Cheung, "A systematic review of blockchain scalability: Issues, solutions, analysis and future research," *Journal of Network and Computer Applications*, vol. 195, p. 103232, 2021.

[13] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, *et al.*, "On scaling decentralized blockchains," in *International Conference on Financial Cryptography and Data Security*, pp. 106–125, Springer, 2016.

[14] G. Danner, I. Hegedűs, and M. Jelasity, "Improving gossip learning via limited model merging," in *International Conference on Computational Collective Intelligence*, pp. 351–363, Springer, 2023.

[15] W. Bi, H. Yang, and M. Zheng, "An accelerated method for message propagation in blockchain networks," *arXiv preprint arXiv:1809.00455*, 2018.

[16] M. A. Imtiaz, D. Starobinski, and A. Trachtenberg, "Empirical comparison of block relay protocols," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 3960–3974, 2022.

[17] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, 2020.

[18] K. Ayinala, B.-Y. Choi, and S. Song, "Pichu: Accelerating block broadcasting in blockchain networks with pipelining and chunking," in *2020 IEEE International Conference on Blockchain (Blockchain)*, pp. 221–228, 2020.

[19] H. Baniata, A. Anaqreh, and A. Kertesz, "Dons: Dynamic optimized neighbor selection for smart blockchain networks," *Future Generation Computer Systems*, vol. 130, pp. 75–90, 2022.

[20] X. Zhang, W. Xia, X. Wang, J. Liu, Q. Cui, X. Tao, and R. P. Liu, "The block propagation in blockchain-based vehicular networks," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8001–8011, 2022.

[21] K. Wang and H. S. Kim, "Fastchain: Scaling blockchain system with informed neighbor selection," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 376–383, 2019.

[22] Y. Aoki and K. Shudo, "Proximity neighbor selection in blockchain networks," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 52–58, IEEE, 2019.

[23] C. Santiago and C. Lee, "Accelerating message propagation in blockchain networks," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 157–160, IEEE, 2020.

[24] J. Wang, "Txilm: Lossy block compression with salted short hashing," June 21 2022. US Patent 11,368,286.

[25] R. Lewis, *Guide to graph colouring*. Springer, 2021.

[26] P. Ayegba, J. Ayoola, E. Asani, and A. Okeyinka, "A comparative study of minimal spanning tree algorithms," in *2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)*, pp. 1–4, IEEE, 2020.

[27] M. Pasetti, E. Sisinni, P. Ferrari, P. Bellagente, and D. Zaninelli, "Comprehensive evaluation of lossless compression algorithms in a real use case for smart grid applications," *Sustainable Energy, Grids and Networks*, vol. 36, p. 101238, 2023.

[28] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[29] H. Yang, G. Qin, and Y. Hu, "Compression performance analysis of different file formats," *arXiv preprint arXiv:2308.12275*, 2023.

[30] Y. Collet, "Rfc 8878: Zstandard compression and the'application/zstd'media type," 2021.