Fully Decentralised Consensus for Extreme-scale Blockchain

Siamak Abdi* Giuseppe Di Fatta[†] Atta Badii[‡]
Giancarlo Fortino[§]

July 2025

Abstract

Blockchain is a decentralised, immutable ledger technology that has been widely adopted in many sectors for various applications such as cryptocurrencies, smart contracts and supply chain management. Distributed consensus is a fundamental component of blockchain, which is required to ensure trust, security, and integrity of the data stored and the transactions processed in the blockchain. Various consensus algorithms have been developed, each affected from certain issues such as node failures, high resource consumption, collusion, etc. This work introduces a fully decentralised consensus protocol, Blockchain Epidemic Consensus Protocol (BECP), suitable for very large and extreme-scale blockchain systems. The proposed approach leverages the benefits of epidemic protocols, such as no reliance on a fixed set of validators or leaders, probabilistic guarantees of convergence, efficient use of network resources, and tolerance to node and network failures. A comparative experimental analysis has been carried out with traditional protocols including PAXOS, RAFT, and Practical Byzantine Fault Tolerance (PBFT), as well as a relatively more recent protocol such as Avalanche, which is specifically designed

^{*}Free University of Bolzano, Italy. Email: sabdi@unibz.it

[†]Free University of Bolzano, Italy. Email: giuseppe.difatta@unibz.it

[‡]University of Reading, United Kingdom. Email: atta.badii@reading.ac.uk

[§]University of Calabria, Italy. Email: giancarlo.fortino@unical.it

for very large-scale systems. The results illustrate how BECP outperforms them in terms of throughput, scalability and consensus latency. BECP achieves an average of 1.196 times higher throughput in terms of consensus on items and 4.775 times better average consensus latency. Furthermore, BECP significantly reduces the number of messages compared to Avalanche. These results demonstrate the effectiveness and efficiency of fully decentralised consensus for blockchain technology based on epidemic protocols.

1 Introduction

Blockchain is a Distributed Ledger Technology (DLT) that stores transactions or any items in the form of sequential blocks. A distributed ledger ensures that data is replicated across multiple nodes for transparency and security. Each block contains a hash of the previous block, creating an immutable chain of blocks. Any blockchain network relies on a consensus mechanism that ensures agreement on the states of certain data or items among distributed participants [6]. For example, in cryptocurrencies like Bitcoin and Ethereum, blockchain technology is utilized to validate and record transactions securely and transparently across a decentralised network of nodes. Consensus in blockchain generally refers to the process of agreeing on the validity of transactions and ensuring the integrity and security of the blockchain. Achieving consensus among nodes in a distributed network is one of the critical challenges in blockchain systems [17], [7].

While there are several consensus algorithms available, such as Paxos [10], Raft [15], Practical Byzantine Fault Tolerance (PBFT) [13], Proof-of-x (e.g. Proof-of-Work (PoW), Proof-of-Stake (PoS)), and Avalanche [16], each has limitations that make them less ideal for certain use cases. For example, traditional protocols Paxos, Raft, and PBFT are effective in closed or permissioned networks but unsuitable for open or public networks. These protocols rely on the existence of a centralized entity or leader, and assume a fully connected communication network, where nodes know and trust each other and can communicate directly with one another (one-to-all and all-to-one). Having a centralized leader makes the system vulnerable to node failures as it is a bottleneck and a single point

of failure in the system, and requires all nodes to trust the leader [3]. Furthermore, it is unreasonable to expect that each node can communicate directly with all other nodes in a large-scale dynamic network. Reaching global consensus in a fully decentralised fashion within a distributed system without full connectivity is an interesting and important challenge [4].

Consensus mechanisms have shifted from centralized models to distributed and decentralised ones [17] in order to overcome these limitations and meet the need for scalability, fault tolerance, availability, data replication, and reliability of data. The removal of a centralized authority or leader from the consensus process also reduces system vulnerability and mitigates the impact of malicious nodes.

Proof-of-x-based algorithms are well-known decentralised consensus mechanisms, but they have their own problems. PoW consumes a lot of resources and energy and may exhibit slowness and inefficiency. Accordingly, they are not suitable for some application domains such as Internet-of-Thing (IoT) due to their high resource consumption [17]. In PoS, another widespread consensus mechanism, validator nodes hold and stake tokens for the privilege of earning transaction fees. PoS can be susceptible to problems related to centralization and to attacks by groups of wealthy nodes. This makes this mechanism more vulnerable to corruption or collusion [7].

Avalanche [16] is a relatively recent decentralised consensus protocol, which works with the combination of a node sampling method and an epidemic diffusion approach to provide a consensus mechanism in a distributed network. Sampling-based algorithms like Avalanche face challenges in achieving global consensus or convergence and often exhibit significant network overhead in large networks. For example, in Avalanche nodes are required to inquire their neighbours to get informed of their votes repeatedly. This approach causes two contrasting problems in the protocol as it is difficult to set the sample size parameter to an appropriate number of neighbours: small values can significantly increase the latency of consensus in large networks and adopting large values may overwhelm the network with messages.

To address the limitations of existing consensus algorithms, we introduce Blockchain

Epidemic Consensus Protocol (BECP), a consensus approach based on epidemic protocols. BECP leverages a randomised communication and computation methodology. The novelty of this work is that unlike the proof-of-based algorithms, which use Gossiping solely for informing participating nodes about new blocks (information dissemination), our method extends its utilization for achieving consensus by means of fully decentralised data aggregation. This approach offers outstanding scalability and robust fault tolerance, and inherits the fast (logarithmic) convergence property of epidemic communication with acceptable cost. Epidemic approaches are leaderless, so there is no bottleneck in the system, and they provide a strong probabilistic guarantee that every participating node will eventually receive the required information and converge to consensus [1], [3].

The rest of the paper is outlined as follows. Section 2 reviews related works of existing consensus protocols in blockchain networks. Section 3 describes the details of the BECP consensus method. Section 4 delves into the implementation of the studied protocols, along with comprehensive network configuration details. Section 5 defines the performance evaluation methodology used for evaluating the performance of BECP. Section 6 discusses the results and findings. Finally, conclusions and future work are concluded in Section 7.

2 Related Work

A consensus algorithm is defined as a protocol or mechanism that is used to achieve agreement among the nodes on a particular item in a distributed network [7]. An effective and inclusive consensus algorithm involves all participants in making decisions based on conflicts within blockchain networks [17]. To date, numerous consensus algorithms have been proposed; the choice of underlying consensus mechanism depends on the type of blockchain and how it is used [17]. Generally, we can classify consensus mechanisms into three broad categories: direct communication-based, proof-of-x-based, and epidemic-based.

Direct communication-based (traditional) consensus algorithms were designed to ad-

dress the challenge of achieving consensus among a group of nodes in a closed or permissioned network where nodes know each other and any new node is verified before joining the network. In such a network, nodes may experience failures or exhibit malicious behaviour, which makes consensus difficult to achieve.

Paxos is a consensus algorithm that was first introduced by Lamport in 1998 [10]. It is a leaderless and three-phase algorithm in which all nodes have the same role and communicate with each other to reach a consensus on a value. Raft is another consensus algorithm that was designed to improve upon the problems of Paxos. It was introduced by Ongaro and Ousterhout in 2014 [15]. PBFT algorithm is one of this class of consensus algorithms that can tolerate byzantine (malicious) behaviours in a distributed system which constitutes 3f + 1 nodes with f malicious nodes. In other words, a consensus can be reached as long as no less than 2f + 1 or 66% of non-byzantine nodes are functioning normally [13].

Paxos, Raft, and PBFT are highly influential algorithms in distributed networks and have served as the basis for many other consensus algorithms. Several other algorithms such as MultiPaxos [5], Fast Paxos [11], Byzantine Paxos [12], Delegated Byzantine Fault Tolerance (DBFT), and Federated Byzantine Agreement (FBA) [17] based on these foundational algorithms have been developed and are currently utilized in blockchain systems to ensure agreement among nodes or participants [14], [6], [17].

Blockchain networks utilize proof-of-x-based consensus algorithms to ensure that all nodes reach a consensus on the state of the network, including the order of transactions and the balance of accounts. Unlike traditional algorithms that have a centralized authority, these algorithms are decentralised. The algorithms reach an agreement by proofing mechanisms and solve inconsistency or forks—instances where two or more conflicting blocks emerge simultaneously—with approaches like selecting the longest chain (Nakamoto protocol) or choosing the heaviest chain in the GHOST protocol [17], [6], [16]. The two most popular consensus algorithms in the Proof-based category used in blockchain are PoW and PoS [8], [18]. An emerging category of consensus algorithms used in distributed ledger systems such as blockchain networks is epidemic-based consensus

algorithms. The 'snow' family consensus algorithms [16] belong to this class which provide a decentralised consensus mechanism with epidemic communication and a sampling method. These algorithms are performed by sampling or inquiring from a small set of participants (neighbours) and reaching an agreement based on predefined measurements. Avalanche is the latest member of the snow family that is utilized in blockchain networks.

On the general problem of agreement and consensus in distributed systems, several works have adopted an epidemic approach.

In [1], the authors propose a novel protocol named Phase Transition Protocol (PTP) to achieve global consensus on the convergence of a distributed information dissemination process. The protocol uses an epidemic approach to exchange information between random nodes and employs a local computation to achieve distributed consensus.

In [2], the authors introduce the Epidemic Consensus Protocol (ECP), which is employed to achieve consensus in distributed data aggregation. ECP achieves consensus by iteratively exchanging information between random nodes until all nodes converge to a common state with explicit local detection of global convergence.

The existing works introduced on consensus mechanisms have some disadvantages that make them less suitable for application in blockchain networks. Traditional algorithms such as PAXOS, RAFT, and PBFT are deterministic, reaching consensus in predefined phases of message passing. However, they are practical only in closed, private, or permissioned blockchain networks. Additionally, the existence of a centralized entity, like a leader, makes them inconsistent with the primary objectives of blockchain networks. Proof-based mechanisms address the centralization problem of traditional algorithms but face challenges such as high computational requirements (PoW) or vulnerability to collusion (PoS).

The recent epidemic-based consensus algorithm, Avalanche, offers a more affordable and reliable decentralised mechanism, but it suffers from a significant volume of sent messages due to its sampling approach. To address these gaps, we introduce a fully decentralised consensus protocol leveraging the advantages of epidemic communication. It employs lightweight local calculations instead of a sampling approach, providing a

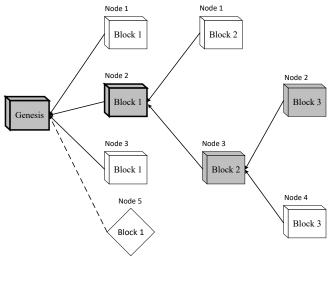
well-suited consensus protocol for blockchain networks.

3 The Consensus Protocol

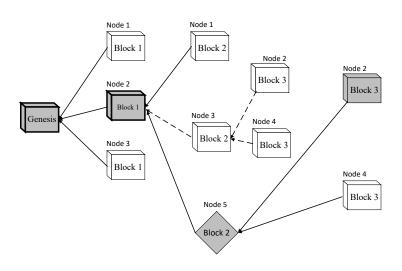
The proposed method is inspired by the works [1], [2], [9], and [3] to devise a fully decentralised consensus protocol that is suitable for the distributed computation in blockchain networks with no reliance on a fixed set of validators or leaders, which also provides solid probabilistic guarantees of convergence and has efficient use of network resources. In the remainder of this section, the details of the proposed algorithm are described. The Blockchain Epidemic Consensus Protocol (BECP) is a fully decentralised epidemic consensus approach applied to blockchain technology. BECP consists of three interrelated protocols: the System Size Estimation Protocol (SSEP), the Node Cache Protocol (NCP), and the Phase Transition Protocol (PTP). SSEP continuously monitors the system size, offering the function qetSystemSize(), which accurately estimates the number of participating nodes in a blockchain system in real time. NCP provides a scalable membership sampling function, getRandomNode(), by randomly selecting nodes for epidemic communication purposes without complete knowledge of the system, which can be dynamic. Finally, the PTP operates as a decentralised consensus algorithm, making use of the other two protocols and resolving issues related to duplicates and guarantees unique IDs in blocks.

The original PTP protocol in [1] has been modified and adapted to work with blockchain. Utilizing the original PTP with the current algorithm does not apply to blockchain networks directly because of the nature and structure of blockchain; in a blockchain, each item or block on which consensus occurs should refer to its parent (previous block). After generating a new block, its content and reference cannot be changed since they are encrypted.

The problem in the original PTP protocol arises since items are generated and accepted without having references. However, in one approach, if participants generate their blocks by referring to their last generated blocks, there is a possibility of breaking the



(a) Case I



(b) Case II

Figure 1: Illustration of Duplicate Block Resolution (Case I) and Backward Scenario (Case II): White blocks represent candidate blocks that dropped, pale blocks denote the currently preferred blocks, the diamond block indicates the newly received block, and blocks with highlighted edges signify confirmed blocks. The text over the blocks indicates the node that created each block.

chain of references when a block is accepted among multiple candidate blocks with similar IDs. An alternate solution to tackle this issue is that nodes should wait for the confirmation of the last block and then generate their new blocks by referring to that block. Once a candidate block is chosen as a confirmed block among other candidate blocks, others will be eliminated from the local caches since they are not valid anymore. This approach, although it guarantees a chain of blocks with correct references, is highly inefficient since nodes must wait for the confirmation of blocks. This decreases the throughput of the systems.

To address this issue more efficiently, we propose a new consensus procedure in algorithm 3 for accepting and resolving inconsistent or duplicate blocks, blocks with similar IDs or similar parents. The consensus procedure in BECP consists of two parts: first, duplicate blocks are resolved in the local caches; then, for the selected blocks, the estimation process is performed. Once the state of a block changes to commit, identified through the estimation process, a consensus is reached on the block. The second part, where consensus is reached on the block, is more time-consuming. As depicted in the algorithm, we define a new variable named B_{pref} , which represents the current preferred block, and impose nodes to generate new blocks by referring to that block. In this way, nodes can generate blocks without the need for the confirmation of the last block.

The current preferred block B_{pref} is the block that is chosen as the possible and potential confirmed block among other candidate blocks in the nodes' local block cache. Once a node receives a block that is selected as the new preferred block, it will accept the new block, and discard the current preferred block along with its descendant blocks. (Algorithm 1). Afterwards, the node will generate a new block by referring to the new preferred block as needed. This approach besides guaranteeing a consistent chain of blocks with correct references, increases throughput since nodes do not wait for the confirmation of the last block.

Algorithm 3 is a revised version of the Resolve Duplicate Block ID Procedure, which is used to determine B_{pref} and insert new blocks into local block caches. First, nodes compare the received block with all existing blocks in the local cache to check for dupli-

cates (line 2)—specifically, if there is an unconfirmed block with an identical ID or similar parents. They must resolve this duplication by choosing either the received block τ' or the existing block τ . The first if statement in line 3 of Algorithm 3 checks if τ' is the same as τ . In that case, the procedure drops τ' but uses its pairs (vp, wp, va, and wa) to update the corresponding pairs in τ . Otherwise, the procedure checks if τ' is selected as B_{pref} (line 5 and 6). If this is the case, the procedure executes the backward procedure, removes τ , adds τ' into the local cache, and sets τ' as B_{pref} . τ' is chosen as a preferred block B_{pref} based on the condition $(\tau'.t = \tau.t \text{ and } \tau'.o < \tau.o)$ or $(\tau'.t < \tau.t)$. In other words, a preferred block B_{pref} is chosen based on the block's generation time and the ID of its originators.

If there is no block with an identical ID to the received block τ' , it is added to the local cache (line 14). This addition depends on verifying whether τ' is connected to the current B_{pref} by checking that the creator node of τ' 's parent is the same as the creator node of B_{pref} . Alternatively, this can be done by comparing the hash of block B_{pref} with the hash of τ' 's parent block. This ensures a consistent and ordered chain of blocks.

Figure 1 (case I) illustrates the process of resolving duplicate or inconsistent blocks. All newly generated blocks refer to the current preferred block (pale blocks), and other candidate blocks (white blocks) are dropped. There is always a preferred block in the local cache. However, in the figure, the diamond-shaped block with the height of 1 is received after all other blocks. Since block 1 is confirmed (the block with highlighted edges), nodes drop the received diamond-shaped block. Figure 1 (case II) depicts a scenario where a backwards occurs: Before receiving the diamond-shaped block 2, there was a preferred block (block 2, created by node 3) and its BECPendants in the chain. When the diamond-shaped block is received, nodes execute the backward procedure, removing the previous preferred block (block 2, created by node 3) and its BECPendants, meaning they are not valid anymore, and setting the diamond-shaped block as the new preferred block. Subsequently, new blocks are connected to this new preferred block.

Furthermore, we offer a new code snipped (Algorithm 2) designed for the block generation process, which prevents nodes from double-spending and producing blocks with

identical IDs and creators. In the algorithm, nodes verify that there are no existing blocks with identical IDs created by themselves, and the new block references the current preferred block. This precautionary step helps prevent potential occurrences of double-spending. Subsequently, upon successful generation of a new block, nodes will set the new block as their current preferred block B_{pref} . We assume that communication is reliable and that all nodes are benign.

```
Algorithm 1: Backward Procedure
```

Data: Received Block τ'

Result: Discarding Invalid Blocks

1 Function BACKWARD(block local cache C_b , τ' :

```
children \leftarrow \tau'.getChildren();

foreach child \in children do

BACKWARD (C_b, child);

end

remove \tau' from the C_b;
```

Algorithm 2: Block Generation Procedure

Result: Generate a new block B_i

1 Function GENERATENEWBLOCK():

```
if block local cache C_b does not contain a block with id (B_{pref}.id+1) then

generate a new block B_i add B_i to the children set of B_{pref}
add B_i into the block local cache C_b;

set B_i to B_{pref}
```

4 Implementation and Network Configuration

To implement and evaluate the BECP protocol alongside other studied blockchain consensus protocols in this work, a blockchain simulator is required. The Just Another Blockchain Simulator (JABS) [19] was chosen as the ideal block simulation environment due to its several advantageous features. It simplifies the simulation of blockchain net-

Table 1: Experiment Settings

Parameters	BECP	Avalanche	PBFT	PAXOS	RAFT
D1 (seconds)	0.1	0.1	_	_	_
Network Latency	[0.01, 0.3)	[0.01,0.3)	[0.01, 0.3)	[0.01, 0.3)	[0.01, 0.3)
Cycle Time (seconds)	0.7	0.7	_	_	0.7
K	_	10	_	_	_
α	_	0.8	_	_	_
β_1	_	50	_	_	_
eta_2	_	150	_	_	_
$T_{\rm block}$ (seconds)	10	10	10	10	10
$P_{ m block}$	5%	5%	_	_	_
ϵ_1	0.01	_	_	_	_
$N_{ m cache}$	50	_	_	_	_
ψ	3	_	_	_	_
Timeout Range	_	_	_	_	1.0 to 1.2

works and makes it easier to understand and work with. Additionally, it offers a high degree of customization, allowing us to adjust the simulation parameters to suit our project requirements. It also supports various network configurations, enabling us to explore different scenarios. The simulator is written in Java programming language, providing the advantage of fast execution and usability.

JABS operates as a discrete-event simulator, where events are characterized as messages received within a node. Specifically, events occur when a node receives a message from a peer. The simulation starts by executing initial events generated by the nodes. During this phase, each node generates a block with a timestamp and sends it to its peer. Furthermore, the nodes continue to generate new blocks at fixed intervals of time with a defined probability.

We implemented the Avalanche protocol with a blockchain framework rather than a DAG structure. Each newly generated block in this framework points to the block with the highest ID within the known set of nodes. This approach enables the nodes to form a chain of blocks, facilitating a fair comparison between systems.

We conducted simulations using the PAXOS, RAFT, PBFT, BECP, and Avalanche protocols on the JABS simulator, employing specific parameters. Our simulations com-

prised two main parts. In the first part, the simulations were run for 3600 seconds (one hour) for PAXOS, RAFT, PBFT, and BECP, and other simulations were run for 600 seconds (10 minutes) for BECP and Avalanche due to the complexity of Avalanche's running time. The simulations were run with seed values of 1 to 5, and a block generation interval of 10 seconds. These simulations were conducted on a Linux server with 400GB of RAM. The settings for each experiment are detailed in the following.

All the protocols were simulated on a WAN network topology where message latency was modeled using a uniform probability function with parameters set to a minimum of 0.01 seconds and a maximum of 0.3 seconds. In the protocols BECP, Avalanche, and Raft, nodes are activated in periodic intervals, therefore we defined the parameter cycle time and set it to 0.7 seconds. In the other two protocols, nodes are activated when they receive a message from the leader. Furthermore, due to the structure of BECP and Avalanche, nodes can propose their blocks simultaneously hence we define an initial interval time D_1 for nodes and set it to 0.1 seconds. In Paxos, Raft, and PBFT only the leader node proposes the blocks, so there is no need to define this parameter.

The parameters of BECP were configured as follows. The ϵ_1 value, representing the estimation error threshold, was set at 0.01, and the ψ , the minimum number of consecutive cycles threshold, was configured at 3 cycles. The N_{cache} representing the number of neighbours' IDs stored in the cache was equal to 50. In Avalanche, sample size (K), quorum size threshold (α) , safe early commitment threshold (β_1) , and consecutive counter threshold (β_2) were set to 10, 0.8, 50, and 150.

The block generation process differed between the deterministic protocols (PAXOS, RAFT, and PBFT), and the probabilistic protocols (BECP and Avalanche). In the traditional protocols, initially, a leader generates a new block and starts the consensus process. After reaching a consensus on the block, the leader node can generate another new block. Contrastingly, Avalanche and BECP nodes can propose their blocks within an interval without waiting for the confirmation of preceding blocks, which is aligned more closely with real-world applications. In our simulations, we set the parameters $T_{\rm block}$ (block generation probability) to 10 seconds and

5% respectively for BECP and Avalanche. For the other protocols, we set $T_{\rm block}$ to 10 seconds.

In the simulator, blocks and transactions were sampled from a distribution based on Bitcoin block and transaction sizes. The message size was determined as the cumulative sum of block sizes in the local block cache. Table 1 provides a concise overview of the protocol settings.

Algorithm 3: Revised Resolve Duplicate Block ID Procedure

5 Performance Evaluation

In this section, to assess the performance of BECP, we conduct a thorough performance comparison with existing protocols of PAXOS, RAFT, PBFT, and Avalanche. Our evaluation includes an in-depth interpretation of the results obtained from the experiments. In the following, we define and describe the measurements of Throughput, Scalability, Communication Overhead, and Consensus Latency which are important metrics for any blockchain network.

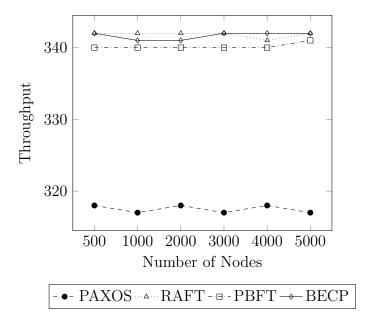


Figure 2: Comparative Throughput Analysis between Traditional Protocols (PAXOS, RAFT, PBFT) and BECP: Impact of Node Scalability (N=500 to 5000) over a Simulation Time of 3600 Seconds

Throughput: Throughput stands out as an important metric for blockchain networks, representing the number of items accepted by nodes within a given time frame. A higher value of Throughput represents the effectiveness of the associate consensus mechanism. An item is defined as a block in the protocols PAXOS, RAFT, PBFT, and BECP and a transaction in the protocol Avalanche based on their structures.

Scalability: Scalability is another crucial metric for blockchain networks. Scalability determines how well a protocol can perform in large-scale networks. Figure 2 and Figure 3 illustrate a comparison of throughput and scalability metrics among the protocols. Figure 2 depicts the throughput comparison for system sizes ranging from 500 to 5000 nodes among the traditional protocols and BECP. RAFT, PBFT, and BECP exhibit the highest throughput, highlighting the efficiency of these protocols. Figure 3 presents a similar comparison between Avalanche and BECP, showing that BECP achieves higher throughput. Considering that Avalanche utilizes transactions instead of blocks, and each block can contain nearly 2000 transactions, BECP demonstrates greater efficiency compared to Avalanche.

Communication Overhead: Communication Overhead refers to the traffic of the network or the total number of sent messages by nodes in a given amount of time. This

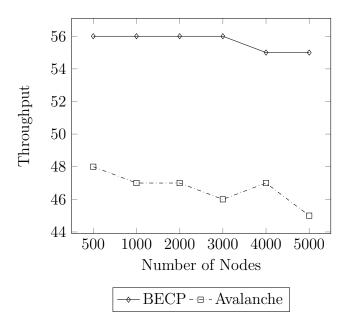


Figure 3: Comparative Throughput Analysis between BECP and Avalanche: Impact of Node Scalability (N = 500 to 5000) over a Simulation Time of 600 Seconds (blocks for BECP, transactions for Avalanche)

metric is important since network congestion increases the delay of sent messages, including block propagation, and as a result, reduces the efficiency of the protocol. Figure 4 and Figure 5 illustrate the comparison of the communication overhead metric of the protocols. According to Figure 4, PBFT demonstrates the highest communication overhead, while PAXOS exhibits the lowest. BECP and RAFT show nearly the same level of communication overhead. However, a comparison between BECP and Avalanche reveals that BECP has lower communication overhead, as shown in Figure 5. For all protocols, communication overhead increases as the number of nodes (system size) increases. However, for PBFT and Avalanche, this increase is more significant.

Consensus Latency: Consensus latency is defined as the time from block generation to acceptance. A protocol with lower consensus latency is considered a desirable protocol. In the experiments, we considered an average of consensus latency for the confirmed blocks. Figure 6 and Figure 7 illustrate the comparison of the average consensus latency metric among the studied protocols. As evident in the figures, PAXOS, RAFT, and PBFT exhibit very small latency compared to BECP and Avalanche. However, BECP demonstrates lower latency compared to Avalanche.

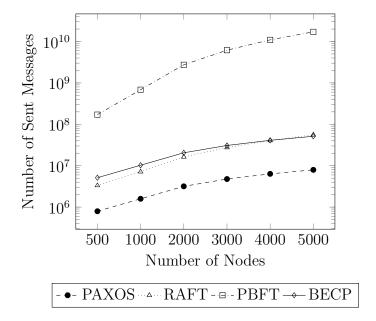


Figure 4: Comparative Communication Overhead Analysis between Traditional Protocols (PAXOS, RAFT, PBFT) and BECP: Impact of Node Scalability (N=500 to 5000) over a Simulation Time of 3600 Seconds

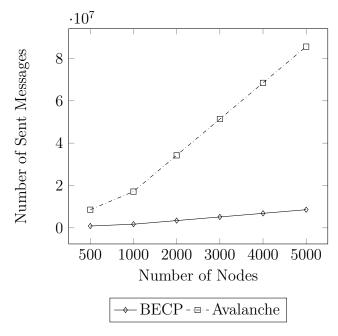


Figure 5: Comparative Communication Overhead Analysis between BECP and Avalanche: Impact of Node Scalability (N=500 to 5000) over a Simulation Time of 600 Seconds

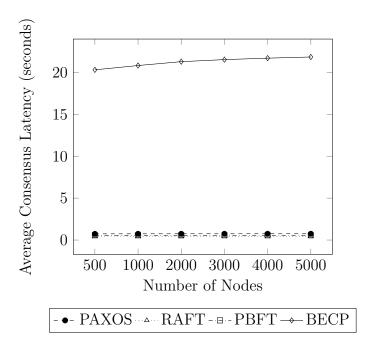


Figure 6: Comparative Consensus Latency Analysis between Traditional Protocols (PAXOS, RAFT, PBFT) and BECP: Impact of Node Scalability (N=500 to 5000) over a Simulation Time of 3600 Seconds

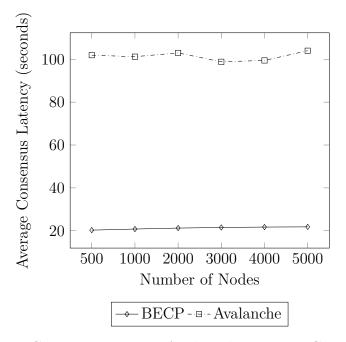


Figure 7: Comparative Consensus Latency Analysis between BECP and Avalanche: Impact of Node Scalability (N=500 to 5000) over a Simulation Time of 600 Seconds

6 Discussion

In this section, we provide a comprehensive discussion of the presented results. According to Figure 2, protocols RAFT, PBFT, and BECP achieved a nearly similar throughput rate, while the protocol PAXOS achieved a slightly lower rate. This disparity can be attributed to the structure of PAXOS, where the promise phase is performed at every round of the consensus process. This frequent promise phase brings latency into the process, resulting in a lower throughput rate. In contrast, algorithm RAFT mitigates this issue by selecting a leader only once, leading to a higher throughput rate.

Figure 3 demonstrates the efficiency of BECP over Avalanche. BECP achieves a slightly higher throughput rate due to its consensus mechanism, which leverages an epidemic diffusion process and local estimations for quicker consensus. During each cycle, nodes simultaneously send and receive information, further boosting performance.

When comparing the communication overhead of the protocols (Figures 4 and 5), we found that PBFT exhibits the highest overhead among the others due to its communication complexity. This overhead primarily stems from its communication complexity, which necessitates a broadcast of votes from all nodes at every phase of the protocol.

Furthermore, protocol RAFT incurs a higher overhead compared to PAXOS. This is because in RAFT, in addition to communication messages for the consensus process, the leader node also broadcasts regular heartbeat messages in the network to maintain its leadership. Overall, PAXOS and RAFT protocols exhibit lower communication overhead due to their centralized structure, where only the leader node sends the blocks. In contrast, protocols BECP and Avalanche demonstrate higher communication overhead compared to PAXOS and RAFT. In these protocols, nodes can generate new blocks with a probability without waiting for the confirmation of previous ones. On the other hand, by comparing the communication overhead of BECP and Avalanche, it is evident that BECP has a lower overhead compared to Avalanche because, in Avalanche, each node draws samples from K neighbours. In contrast, nodes in BECP send a message to only one neighbour.

The traditional protocols exhibit very low average consensus latency (Figure 6) due

to their deterministic structure. They can achieve consensus on an item in three message passes, apart from the leader election process. However, this approach involves achieving consensus on blocks one after another. BECP and Avalanche demonstrate higher average consensus latency (Figure 7) because, unlike traditional protocols, they do not rely on the direct communication assumption. This approach closely resembles real-world applications, where users can generate blocks with a probability. However, as depicted in Figure 7, BECP has lower latency compared to Avalanche, while achieving the same throughput rate as the traditional protocols. This efficiency is due to its epidemic consensus mechanism, which broadcasts blocks throughout the network, resulting in faster consensus.

Based on the results obtained from 500 to 5000 nodes, we found that the protocol BECP demonstrates desirable scalability properties compared to the other tested protocols. BECP maintains a steady rate of throughput, which surpasses Avalanche's rate of throughput. Additionally, it exhibits lower communication overhead compared to PBFT and Avalanche.

7 Conclusion

In this paper, the Blockchain Epidemic Consensus Protocol (BECP) protocol has been introduced. BECP is a novel and fully decentralised consensus system for blockchain networks, that leverages epidemic communication and local computation to facilitate consensus on blocks. Unlike traditional protocols such as PAXOS, RAFT, and PBFT, BECP operates without a designated leader, thereby ensuring robust decentralisation. Unlike PoW mechanisms, BECP imposes minimal resource demands, and unlike PoS, it is not vulnerable to collusion. Our simulations and analyses confirm that BECP shows favourable metrics including throughput, scalability, communication overhead, and consensus latency when compared to existing protocols. Future efforts will focus on augmenting BECP consensus mechanism to include the ability to detect node failures and trigger a recovery mechanism to provide system resilience towards these failures.

References

- [1] Mosab Ayiad, Amogh Katti, and Giuseppe Di Fatta. Agreement in epidemic information dissemination. In *Internet and Distributed Computing Systems: 9th International Conference*, IDCS 2016, Wuhan, China, September 28-30, 2016, Proceedings 9, pages 95–106. Springer, 2016.
- [2] Mosab M Ayiad and Giuseppe Di Fatta. Agreement in epidemic data aggregation. In 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS), pages 738–746. IEEE, 2017.
- [3] Francesco Blasa, Simone Cafiero, Giancarlo Fortino, Giuseppe Di Fatta, et al. Symmetric push-sum protocol for decentralised aggregation. In *Proc. of the Int. l Conf. on Advances in P2P Systems*, pages 27–32, 2011.
- [4] Daniel Cason, Nenad Milosevic, Zarko Milosevic, and Fernando Pedone. Gossip consensus. In Proceedings of the 22nd International Middleware Conference, pages 198–209, 2021.
- [5] Saksham Chand, Yanhong A Liu, and Scott D Stoller. Formal verification of multipaxos for distributed consensus. In *International Symposium on Formal Methods*, pages 119–136. Springer, 2016.
- [6] Md Sadek Ferdous, Mohammad Jabed Morshed Chowdhury, Mohammad A Hoque, and Alan Colman. Blockchain consensus algorithms: A survey. arXiv preprint arXiv:2001.07091, 2020.
- [7] Ziad Hussein, May A Salama, and Sahar A El-Rahman. Evolution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms. *Cybersecurity*, 6(1):30, 2023.
- [8] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols.

 In Secure Information Networks: Communications and Multimedia Security IFIP

- TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS'99) September 20–21, 1999, Leuven, Belgium, pages 258–272. Springer, 1999.
- [9] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In 44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings., pages 482–491. IEEE, 2003.
- [10] Leslie Lamport. Paxos made simple. ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001), pages 51–58, 2001.
- [11] Leslie Lamport. Fast paxos. Distributed Computing, 19:79–103, 2006.
- [12] Leslie Lamport. Byzantizing paxos by refinement. In *International symposium on distributed computing*, pages 211–224. Springer, 2011.
- [13] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pages 203–226. 2019.
- [14] Bahareh Lashkari and Petr Musilek. A comprehensive review of blockchain consensus mechanisms. *IEEE access*, 9:43620–43652, 2021.
- [15] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In 2014 USENIX annual technical conference (USENIX ATC 14), pages 305–319, 2014.
- [16] Team Rocket. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies. *Available [online].[Accessed: 4-12-2018]*, 2018.
- [17] Arshdeep Singh, Gulshan Kumar, Rahul Saha, Mauro Conti, Mamoun Alazab, and Reji Thomas. A survey and taxonomy of consensus protocols for blockchains. *Journal* of Systems Architecture, 127:102503, 2022.
- [18] Wai Yan Maung Maung Thin, Naipeng Dong, Guangdong Bai, and Jin Song Dong. Formal analysis of a proof-of-stake blockchain. In 2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS), pages 197–200. IEEE, 2018.

[19] Habib Yajam, Elnaz Ebadi, and Mohammad Ali Akhaee. Jabs: A blockchain simulator for researching consensus algorithms. IEEE Transactions on Network Science and Engineering, 2023.