mathematics

MDPI

Article

Blizzard: A Distributed Consensus Protocol for Mobile Devices

Mehrdad Kiamari [1,*], Bhaskar Krishnamachari [1,2] and Seokgu Yun [3]

[1] Department of Electrical and Computer Engineering, University of Southern California,
Los Angeles, CA 90089, USA; bkrishna@usc.edu
[2] Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA
[3] SovereignWallet Network, Seoul 442736, Republic of Korea; phantom@sovereignwallet.network
* Correspondence: kiamari@usc.edu

Abstract: We present Blizzard, a Byzantine fault tolerant (BFT) distributed ledger protocol that is aimed at making mobile devices first-class citizens in the consensus process. Blizzard introduces a novel two-tier architecture by having the mobile nodes communicate through online brokers, and includes a decentralized matching scheme to ensure each node connects to a certain number of random brokers. Through mathematical analysis, we derive a guaranteed safety region (i.e., the set of ratios of malicious nodes and malicious brokers for which the safety is assured) for the Blizzard protocol. Liveness is shown as well. We analyze the performance of Blizzard in terms of its throughput, latency, and message complexity. Through experiments based on a software implementation, we show that Blizzard is capable of throughput on the order of several thousand transactions per second per shard and sub-second confirmation latency.

Keywords: mobile-based distributed consensus; transactions; safety; latency; throughput

MSC: 68W15

check for updates

Citation: Kiamari, M.; Krishnamachari, B.; Yun, S. Blizzard: A Distributed Consensus Protocol for Mobile Devices. *Mathematics* **2024**, *12*, 707. https://doi.org/10.3390/math12050707

Academic Editor: Hsien-Chung Wu

Received: 16 November 2023
Revised: 14 February 2024
Accepted: 19 February 2024
Published: 28 February 2024

1. Introduction

It has been estimated that there are more than 4 billion mobile device users worldwide [1], and the mobile ecosystem is still growing [2]. We are therefore motivated to examine the design of a mobile-first consensus protocol suited for distributed ledger maintenance that can leverage such a massive user base. Specifically, unlike previous research focusing on security and privacy concerns related to using mobile devices for client software hosting (such as digital wallets) that merely send and receive transactions [3,4], our study seeks to investigate the potential for these devices to engage more actively in the consensus mechanism by participating in transaction validation. While a major reason for previous work limiting the role of mobile devices to that of a client is their generally more resource-constrained nature compared to larger compute servers, it is important to note that today's mobile devices are already capable of significant computation, communication, and storage. Furthermore, as Moore's law has been shown to apply to mobile platforms as well [5], we can expect these capabilities to continue expanding in the future.

The distributed mobile-based consensus protocol for distributed ledger maintenance that we propose is called Blizzard. Blizzard (the shorter version of this paper is presented in [6]) is a leaderless consensus protocol in which mobile nodes connect and communicate with a number of online servers called brokers. Mobile nodes only need to store and communicate with a number of end addresses that scales with the number of brokers in the system, rather than with the number of all mobile devices in the system. Of course, non-mobile devices such as online servers could also participate as validators, our point is that this is the first protocol to explicitly allow for mobile device-based validators, which can be switched off or connect intermittently. Each mobile node connects to a random subset of these servers for a given period of time and communicates with all other mobile nodes in each broker's group. Effectively, each broker creates a broadcast group of mobile nodes that can query and respond to each other. Importantly, Blizzard is designed to operate effectively in environments with heterogeneous mobile devices, seamlessly adapting to varying levels of operating systems, hardware capabilities, and communication technologies among these devices.

We briefly enumerate our contributions in this work as follows:

1. Blizzard is the first mobile-based leaderless Byzantine fault tolerant (BFT) distributed consensus protocol. Not only can mobile devices issue transactions, but they can participate in the core transaction verification and consensus process as well. This could increase the number of nodes capable of participating in validation by 2 to 3 orders of magnitude, thereby enhancing both adoption and security.
2. We propose a novel two-tier protocol, where consensus between mobile nodes is enabled by the use of online brokers, and a decentralized matching scheme that ensures each mobile node connects to exactly k random brokers.
3. Provable safety guarantee. We mathematically derive the set of ratios of Byzantine nodes and brokers for which Blizzard's safety is guaranteed. We also discuss why liveness holds in Blizzard.
4. We analytically characterize the throughput of Blizzard by modeling the sequential pipeline involved in processing transactions at each node and identifying the throughput bottleneck via empirical profiling.
5. Likewise, we also analytically characterize the confirmation latency and message complexity of Blizzard. We show that the use of brokers in Blizzard creates a communication topology that allows for efficient consensus; under reasonable parameter settings, transactions are propagated in Blizzard within just four communication rounds with high probability.
6. Through experiments based on a software implementation of Blizzard, we show that it is capable of 10,000 transactions per second per shard, and allows for sub-second confirmations.

2. Related Works

The original Bitcoin paper by Nakomoto [7] provided a joint solution to several problems, including consensus (through longest-chain adoption), ledger representation (hashed chain of blocks), Sybil control (through proof of work), and incentives (through mining and transaction rewards). However, we argue that it is both possible and valuable to consider these various components separately. To place our work in context, we briefly survey the literature with respect to these five dimensions.

Consensus: In this work, we focus solely on the problem of distributed consensus, focusing, in particular, on BFT. In [8], the Bitcoin protocol is formalized, and it is shown how it can be extended to provide Byzantine agreement. There is, in fact, an earlier study on BFT consensus for distributed systems, primarily focused on leader-based protocols, such as PBFT [9], BFT-Smart [10], and others. More recently, in the context of Blockchain, several new leader-based consensus solutions have been proposed that improve the speed of BFT consensus under partial synchrony assumptions. These include Tendermint [11], Hotstuff [12], and Casper CBC [13]. There has also been recent work on leaderless protocols, including Hashgraph [14], Avalanche [15], DBFT [16], and Aleph [17], which do not require having a single proposer or leader for each round of consensus. The Blizzard protocol described in this paper is a leaderless BFT protocol that is designed to allow mobile devices to participate in the consensus by leveraging online brokers. Blizzard builds on the idea of gossip-based consensus presented originally in Avalanche but is structurally significantly different due to the introduction of aggregating brokers and, therefore, requires a different safety, throughput, and latency analysis, which are all presented in this work.

Ledger Representation: There are broadly two classes of approaches to ledger representation in distributed ledger systems, either using a linear Blockchain as in the original Bitcoin, Ethereum, and many other protocols, or allowing transactions (or blocks) to point to other previous transactions (or blocks) resulting in a directed acyclic graph (DAG) data structure. Examples of such DAG-based protocols include IOTA [18], Hashgraph [14], Avalanche [15], and Helix [19].

Sybil control: Surveying different Sybil control mechanisms, we find that Ethereum also uses proof of work for Sybil control, while newer projects and systems, such as Cosmos [20], Algorand [21], Ouroboros [22], Dfinity [23], and Ethereum 2.0 [24], have been exploring energy-efficient alternatives such as proof of stake and delegated proof of stake. Meanwhile, in permissioned blockchains, such as Hyperledger Fabric [25] and Hyperledger Sawtooth [26], Sybil control is handled explicitly by only allowing a limited set of pre-vetted, pre-approved validators into the system. In this work, we do not treat the problem of Sybil control, similar to other prior work focused on distributed consensus.

Incentives: Finally, like most prior work on BFT consensus protocols and permissioned blockchains (but unlike many cryptocurrency projects such as Bitcoin and Ethereum), Blizzard is agnostic to how incentives are provided to validating nodes. This allows its use for a broader range of use cases beyond cryptocurrency, but keeps the flexibility to allow incentive mechanisms to be employed as a separate modular layer if needed.

Mobility: In the context of mobile-based consensus protocols, several key studies have laid the groundwork, although they primarily focus on nodes with mobility rather than utilizing smartphones as consensus nodes. An efficient consensus protocol for MANETs, based on a hierarchical approach for message efficiency, is presented by Wu et al. [27]. The consensus problem in mobile environments with disconnections is addressed by Badache et al. [28]. In [29], a modular approach for designing hierarchical consensus protocols in mobile ad hoc networks is proposed. The design of a message-efficient consensus protocol for MANETs is presented in [30]. Our proposed Blizzard protocol, as the first one to specifically use smartphones for consensus, is a significant departure from these earlier works.

One crucial aspect of widely adopted protocols such as Bitcoin, Ethereum, and Avalanche is their open, permissionless nature, allowing any device to join or leave the network at any time. Therefore, they do not utilize any information about the total number of validator nodes when voting for blocks. This significant feature, which is also essential for mobile networks, incurs the cost that such a validator number-agnostic protocol can only be proved to operate safely under a synchronous model [31]. Specifically, the safety of permissionless consensus in partially synchronous or asynchronous models is not guaranteed [31]. On the other hand, the Blizzard protocol, as we introduce, does not make assumptions about the *total* count of nodes involved. Therefore, its safety assurance is limited to a synchronous model, similar to the situation with Bitcoin, Ethereum, and Avalanche.

In Table 1, we summarize some of the main Blockchain protocols and their key properties and attributes, along with Blizzard, to help put our contribution in context. In a nutshell, as we will show, Blizzard provides both high throughput and low latency comparable to state-of-the-art protocols while enabling significantly greater scalability by allowing mobile devices to serve as transaction validators.

Table 1. Comparison of different protocols.

Protocol	Sybil Control Method	Leaderless	Ledger Structure	BFT	Transaction per Second per shard	Confirmation Latency	Number of Validators	Mobile-Based
Bitcoin [7]	PoW	No	Chain	No	7	$\sim$40 min	100k+	No
Ethereum [24]	PoW/PoS	No	Chain	No	20	$\sim$60 s	100k+	No
Tendermint [11]	Agnostic	No	Chain	Yes	$\sim$10,000	$\sim$2–15 s	$\sim$100–1k	No
Avalanche [15]	Agnostic	Yes	DAG	Yes	$\sim$3400	$\sim$1.35 s	100k+	No
Blizzard	Agnostic	Yes	DAG	Yes	$\sim$10,000	$\sim$0.65 s	**100M+**	Yes

3. System Model

We consider a broker-assisted mobile network, where disjoint sets $\mathcal{N}_C$ and $\mathcal{N}_M$, respectively, represent sets of correct and malicious mobile nodes (set of all nodes is denoted by

$\mathcal{N} := \mathcal{N}_C \cup \mathcal{N}_M$). Furthermore, set $\mathcal{B}$ indicates the set of all brokers, which consists of sets $\mathcal{B}_C$ and $\mathcal{B}_M$, representing the subsets of correct and malicious brokers, respectively. Other notations are provided in Table 2.

Table 2. Notation description.

n	:	Number of all mobile nodes (where $n = \vert\mathcal{N}\vert$).
c	:	Number of correct mobile nodes.
b	:	Number of Byzantine mobile nodes.
m	:	Number of all brokers.
m_c	:	Number of correct brokers.
m_b	:	Number of Byzantine brokers.
$\mathcal{N}^{(b)}$	:	Set of connected mobile nodes to broker b.
$\mathcal{B}^{(u)}$	:	Set of connected brokers to mobile node u.
k	:	Number of brokers being sampled by each mobile node.
α	:	Majority threshold of mobile nodes for considering a "yes" vote.
η	:	Majority threshold of brokers for considering a "yes" vote.
β_1	:	Security threshold used for consecutive counter.
β_2	:	Security threshold used for confidence counter.

Mobile nodes issue cryptographically signed transactions. We assume all validating nodes have access to a common function that can determine if any two transactions are conflicting or not (this is general enough, for example, to cover the detection of conflicting transactions under either a UTXO or account-based model). Correct nodes never issue conflicting transactions, while Byzantine nodes may issue conflicting transactions.

Regarding misbehavior acts, we assume the existence of both Byzantine mobile nodes as well as brokers. In Blizzard, malicious brokers are effectively limited to suppressing messages as they do not sign or initiate any messages themselves and are assumed to not be able to forge messages from mobile nodes. As far as the Byzantine behavior for malicious mobile nodes is concerned, malicious nodes are computationally limited (not able to forge signatures), while they can choose any execution strategy that they desire. Moreover, the consensus remains unaffected by mobile nodes turning off as long as the fraction of correctly connected mobile nodes is sufficiently high [15].

Since we aim to have a protocol such that any vote on a new transaction would be a vote on some previous transactions, we incorporate a DAG structure into our protocol. To do so, some parent transactions would be assigned for each new transaction. Therefore, any vote on a specific transaction is a vote on all of its ancestor transactions (i.e., all transactions accessible through the parents of a transaction) as well. The overview of the DAG structure is presented in Appendix B.

We next present how our proposed Blizzard scheme works in detail.

3.1. Proposed Blizzard Scheme

Our proposed scheme works as follows: each node $u \in \mathcal{N}$ connects to k brokers (represented by $\mathcal{B}^{(u)}$) uniformly at random (the mechanism of k random connections will be discussed in the following subsection) and queries them on a new transaction T.

Upon receiving a query, each broker $b \in \mathcal{B}^{(u)}$ computes η-majority vote on T by querying all of its connected nodes denoted by $\mathcal{N}^{(b)}$. Then, each node $v \in \mathcal{N}^{(b)}$ for all brokers $b \in \mathcal{B}^{(u)}$ affirmatively responds to the query if all of the ancestor transactions of T are currently the preferred choice of transaction in their corresponding conflict sets in the stored DAG of node v. Afterwards, broker $b \in \mathcal{B}^{(u)}$ aggregates the count of all positive responses collected from nodes $\mathcal{N}^{(b)}$ and sends an affirmative response to all its connected nodes using a suitable key aggregation scheme if $\geq \eta|\mathcal{N}^{(b)}|$ (where $\frac{1}{2} < \eta < 1$) collected responses from nodes $\mathcal{N}^{(b)}$ are positive. In the case of having $\geq \alpha k$ (where $\frac{1}{2} < \alpha < 1$) positive responses, collected from brokers $\mathcal{B}_u$ for T, then T will be appended to the stored DAG of node u and node u never queries T again. The protocol would continue by following the same process for all nodes having T in their known transaction sets. An example of our proposed scheme for $k = 2$, where one node queries on a new transaction, is depicted in Figure 1. The details of the Blizzard protocol are elaborated in Algorithm 1 and side function $Query_{broker}(.,.)$ in (1).

$$Query_{_{Broker}}(b, T) := \begin{cases} 1 & \text{if } \sum_{u' \in \mathcal{N}^{(b)}} Query_{Node}(u', T) \geq \eta |\mathcal{N}^{(b)}| \\ 0 & \text{if else} \end{cases} \tag{1}$$

where $Query_{Node}(u', T)$ is 1 if transaction T and all its ancestor transactions are the *preferred* transactions among their corresponding *conflict sets* (see Appendix B for more details on *conflict sets* and *preferred* transaction in a conflict set); otherwise, it is 0.

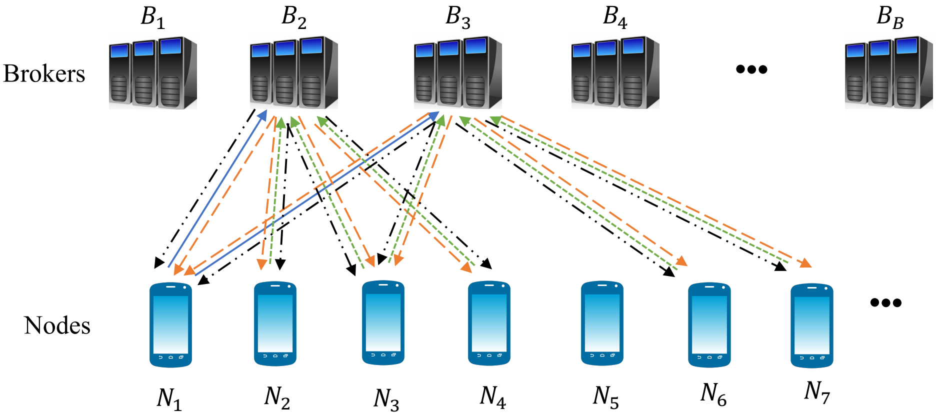

Figure 1. An illustration on how Blizzard works for $k = 2$. Mobile Node 1 queries brokers B_2 and B_3 on a new transaction these brokers have not queried yet (shown with blue arrows). Then, these brokers query all their connected mobile nodes about the transaction (depicted with orange arrows). Afterwards, all connected mobile nodes respond back to queries (shown with green arrows) and brokers reflect the majority vote to all of the connected nodes (presented by dashed-black arrows).

```
Algorithm 1: Blizzard Algorithm
    Input: Set of mobile nodes $\mathcal{N}$, brokers $\mathcal{B}$, and transactions coming over time
    Output:
   Set of non-conflicting transactions accepted by every correct mobile node $u \in \mathcal{N}$
 Initialization:
 - Each node $u \in \mathcal{N}$ randomly connects to $k$ brokers represented by $\mathcal{B}^{(u)}$.
 - Set $\mathcal{T}_u = \mathcal{Q}_u = \emptyset$ for all nodes which do not issue transaction where $\mathcal{T}_u$ and $\mathcal{Q}_u$
   represent known and queried transaction sets of node $u$, respectively.
 while there is a transaction $T$ at any node $u$ such that $T \in \mathcal{T}_u, T \notin \mathcal{Q}_u$ do
    - $R_{\text{brokers}} := \sum_{b \in \mathcal{B}^{(u)}} Query_{Broker}(b, T)$;
    if $R_{brokers} \geq \alpha k$ then
       - $v_{u,T} = 1$ // $T$ receives a voucher and appends to the DAG of node $u$.
       - Update DAG and the conflicting sets of node $u$ after appending $T$.
    end
    - $\mathcal{Q}_u = \mathcal{Q}_u \cup \{T\}$ // mark $T$ as a queried transaction.
 end
```

We next present a distributed mechanism to enforce k random connections for mobile nodes.

3.2. Distributed Random Matching

For Blizzard to work, we need to ensure that each mobile node is connected to k *random* brokers. Since random connection plays a key role in preventing collusion between Byzantine entities, we propose a mechanism which requires all mobile nodes, even Byzantine ones, to provide a proof of their connections being random.

Our proposed distributed random matching scheme works as follows:

1. Each mobile device applies a hash function on the combination of the random number coming from a distributed random beacon (note that a distributed random beacon is now live online at https://drand.love/ (accessed on 15 November 2023)) [32] with the ID of the mobile device. Regarding the hash function, it outputs B bits where 0 and 1 are equally likely. Then, the indices of the first k ones represent the brokers each mobile device has to connect with. The mobile device afterwards sends the output of its hash function as well as its IDs to the brokers it is supposed to connect with.
2. Brokers validate the ID of mobile devices, verify that hash values generated by mobile devices are correctly produced, taking into account the distributed random beacon, and thus verify that mobile devices are authorized to connect.

An illustration of the proposed distributed random matching scheme is depicted in Figure 2. One crucial factor ensuring the effectiveness of our proposed scheme is that the hash function outputs a sufficiently long sequence (or equivalently, a large value of B), ensuring that there are at least k ones in the hashed value with high probability. Theorem 1 determines parameter B for our proposed distributed random matching scheme to satisfy this condition.

Theorem 1. *In order to ensure the probability of existing at least k ones in a sequence of length B to be* $1-\delta$ *(for small* δ*), parameter B should satisfy* $\frac{1}{2}\log\frac{1}{\delta} = (\frac{1}{2} - \frac{k-1}{B})^2 B$.

Proof.

$$P(H(B) \le k-1) \le \delta = \exp\left(-2\epsilon^2 B\right) \tag{2}$$

where (2) follows from Hoeffding's inequality, $H(B)$ indicates the number of ones in a sequence of length B, and $\epsilon := \frac{1}{2} - \frac{k-1}{B}$. □

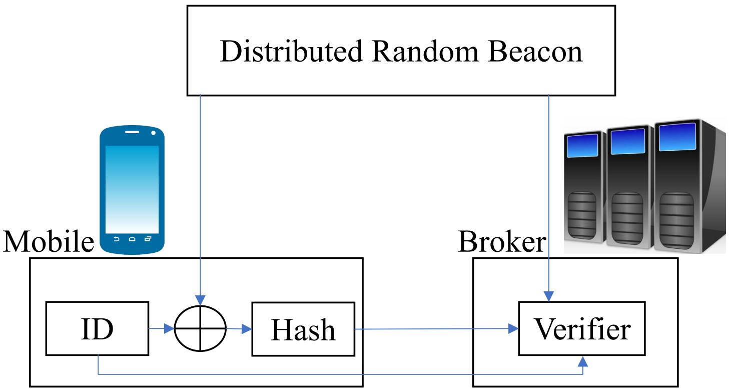

Figure 2. Proposed distributed random matching scheme for connecting each mobile node to k brokers.

4. Safety and Liveness

As mentioned earlier, it has been shown in [31] that permissionless protocols that do not make any assumption about the total number of nodes involved in the consensus protocol can only be proved to operate correctly under a synchronous model. This applies to previously proposed protocols such as Bitcoin, Ethereum, and Avalanche, and also to Blizzard. Thus, in the following discussion, we assume a synchronous network where the maximum latency for any message is bounded by a known constant.

4.1. Safety Analysis

To establish the robustness of the Blizzard scheme in terms of safety, it suffices to demonstrate that every correct node uniformly agrees on the same transaction from a set of conflicting ones within a finite time frame almost surely. For instance, consider a scenario where transaction T_1 is already present and transaction T_2 (which conflicts with T_1) is newly introduced to the network. For simplicity and illustrating the nodes' preferences between two conflicting transactions, we assign two distinct colors to the nodes: red and blue.

Nodes favoring transactions T_1 and T_2 are depicted as red- and blue-colored nodes, respectively. For simplicity and without any loss of generality, our analysis concentrates on scenarios where all correct nodes unanimously choose the red color as their consensus.

The operational guidelines for mobile nodes and brokers in the network are outlined as follows:

1. Every correct mobile node always responds with honesty upon receiving any query.
2. A queried Byzantine node may respond with any color or even refuse to respond.
3. Every correct broker computes η-majority of the votes collected from all nodes connected to it and *broadcasts* the majority vote to *all* of them.
4. Byzantine brokers cannot forge information since they cannot cryptographically sign transactions from nodes. However, a Byzantine broker may compute η-majority of the votes collected from an arbitrary *subset* of nodes connected to it and send the computed vote to any selected nodes that are connected to this broker. They could also choose not to communicate.

The primary objective of adversaries with respect to safety is to prevent correct nodes from reaching a consensus on the same transaction among all conflicting transactions within a finite time. To achieve this, malicious nodes do their best to hinder the process of all correct nodes reaching a unanimous decision on a single color.

In order to prevent adversarial attacks, similar to Avalanche, we incorporate two following counters for each mobile node:

(1) *Conviction in Current Color (C3)* counter to store how many consecutive computed majority votes have resulted in the same color. Once a node flips its color, this counter reset to zero. Furthermore, a node locks into the current color when this counter exceeds some security parameter β_1.

(2) *Confidence* counter to take into account the number of queries that have yielded a majority vote for their corresponding colors. A node flips its color only if the confidence value of its computed vote is larger than the confidence value of the current color. Moreover, a node locks into a color once the confidence value of this color exceeds some security parameter β_2.

Color-based Blizzard: Considering the aforementioned assumptions, the color-based Blizzard scheme works as follows: each mobile node connects to k brokers uniformly at random and queries them. Please note that randomly connecting each mobile to k brokers is guaranteed by the distributed random matching scheme described in the previous section. Then, each of these brokers queries all its connected nodes regarding their colors. Subsequently, each of the connected nodes, upon being queried, sends its color to the brokers that it is connected to. Once the color of all nodes connected to broker i received, then the broker computes whether $\geq \eta|\mathcal{N}^{(i)}|$ (where $\mathcal{N}^{(i)}$ represents nodes connected to broker i and $\eta \in (\frac{1}{2}, 1])$ collected responses have the same color or not. In the case of having $\geq \eta|\mathcal{N}^{(i)}|$ responses with the same color, then the broker broadcasts the computed majority vote to all nodes connected to it.

Once α-majority of votes collected from brokers connected to a node yields to a color, i.e., receiving $\geq \alpha k$ positive responses, the confidence counter for that color will increase by one. If this color (the one computed from the majority votes coming from k brokers) is the same as the node's current color, C3 counter increases by one; otherwise, the node resets C3 counter to zero. A node flips its color to a new color if the confidence counter of new color is larger than the confidence counter of current color.

Safety Analysis of Color-based Blizzard: Without loss of generality, we assume that the initial node colors are randomly assigned such that $\frac{c}{2} + 1$ nodes have red color, while the remaining nodes are blue. This is the worst-case scenario for reaching consensus due to balance in number of nodes with different colors.

Let us represent correct mobile nodes that prefer red and blue colors with u and v, respectively. To show consensus is achieved, we can show that nodes that prefer blue color gradually gain confidence in the red color over time, and they eventually turn into red-preferred nodes with high probability. By defining $z.\text{Conf}[R]$ as the confidence of node z in color R, i.e., the number of queries yielding a majority vote for color R for node z (and similar definition for $z.\text{Conf}[B]$), the rule for color change is straightforward: node z will change its color to red if $z.\text{Conf}[R] > z.\text{Conf}[B]$ and flips to blue otherwise.

Our proof for reaching consensus will be shown as the following steps:

Step 1: After some finite time, the system reaches the point where there are $\frac{c}{2} + \Delta$ red nodes while the remaining nodes are blue.

Step 2: At this point, v nodes have negative average growth in confidence value for blue color at any time (note that average growth in confidence value of node z for color x is $\mathbb{E}[z.\text{Conf}^{\,(t)}[x]] - \mathbb{E}[z.\text{Conf}^{\,(t-1)}[x]]$), with high probability. After a short period of time, we have $v.\text{Conf}[B] - v.\text{Conf}[R] = -1$, with high probability, and as a result, v nodes flip their color to red. Note that u nodes just gain more confidence in red as time elapses in this step.

4.1.1. Step 1

We can model our scheme as a discrete-time Markov Chain with state s_i, $\forall i \in \{0, \dots, c\}$, where s_i represents the state with i red and $c - i$ blue nodes, with transition probability matrix M.

Since each of these transition probabilities is a function of the adversary, let us now elaborate upon its behavior. The most malicious scenario conducted by the adversary aims to achieve the following goal: keep the confidence value of blue and red colors nearly the same for u nodes while letting v nodes increase their confidence value of blue color as much as they can. More formally, the scenario is to have $u.\text{Conf}[R] = u.\text{Conf}[B] + 1$ and maximizing κ, where $\kappa \triangleq v.\text{Conf}[B] - v.\text{Conf}[R]$. Therefore,

- **When a u node queries:** All Byzantine mobile nodes acquire red colors. Regarding the malicious brokers, all of them act with honesty without suppressing any color.
- **When a v node queries:** All Byzantine mobile nodes pick blue colors, and all Byzantine brokers with a red-color majority switch to a blue-majority broker by not reflecting their red-color mobile nodes. With high probability, as shown in Appendix A, a node is connected to at most $f \triangleq \frac{m_b r}{m}$ Byzantine brokers in a population of r red brokers and $m - r$ blue brokers.

Since none of the transition probabilities are zero, the system can reach to the state $s_{c/2+\Delta}$ in finite time. We will discuss how Δ can be determined in the next step. It is important to emphasize that we set the security parameters k, α,η, β_1, and β_2 such that no node finalizes its color during this step.

4.1.2. Step 2

In this step, we show how v nodes flip their color to red, and as a result, all correct nodes reach consensus with high probability. To do so, we write the expected confidence value for mobile nodes at time t as follows:

$$\begin{aligned}
\mathbb{E}[u.\text{Conf}^{\,(t)}[R]] &= \mathbb{E}[u.\text{Conf}^{\,(t-1)}[R]] + P(\mathcal{C}_u^{(t)} \text{is red}) \\
&= \mathbb{E}[u.\text{Conf}^{\,(t-1)}[R]] + \sum_{r=\alpha k}^{m} P(\mathcal{C}_u^{(t)} = \text{red} \mid \mathcal{A}_r^i) P(\mathcal{A}_r^i) \\
&= \mathbb{E}[u.\text{Conf}^{\,(t-1)}[R]] + \sum_{r=\alpha k}^{m} \Big(\sum_{j'=\alpha k}^{k} \frac{\binom{r}{j'}\binom{m-r}{k-j'}}{\binom{m}{k}} \Big) \binom{m}{r} p_{i+b}^{r} (1 - p_{i+b})^{m-r},
\end{aligned} \tag{3}$$

where $\mathcal{C}_u^{(t)}$ represents the color of the computed vote of node u at time t. Moreover, $\mathcal{A}_r^i$ denotes the event of having r red-majority brokers and $m - r$ blue-majority brokers when there are i red mobile nodes in a population of n nodes. Therefore, $P(\mathcal{A}_r^i) = \binom{m}{r} p_i^r (1-p_i)^{m-r}$, where p_i, the probability of a broker to be red given total i red nodes in a population of n nodes, is

$$p_i \triangleq \sum_{\ell=1}^{n} \underbrace{\frac{\sum_{j \geq \eta\ell} \binom{i}{j}\binom{n-i}{\ell-j}}{\binom{n}{\ell}}.}_{\text{probability of the broker being red given } \ell \text{ connections}} \times \underbrace{\binom{n}{\ell}(\frac{1}{m})^{\ell}(1-\frac{1}{m})^{n-\ell}}_{\text{probability of the broker having } \ell \text{ connections}} \tag{4}$$

Similarly, we would have

$$\mathbb{E}[u.\text{Conf}^{\,(t)}[B]] = \mathbb{E}[u.\text{Conf}^{\,(t-1)}[B]] + \sum_{r=\alpha k}^{m} \Big(\sum_{j'=\alpha k}^{k} \frac{\binom{r}{j'}\binom{m-r}{k-j'}}{\binom{m}{k}} \Big) \binom{m}{r} (1-p_i)^r {p_i}^{m-r} \tag{5}$$

$$\mathbb{E}[v.\text{Conf}^{\,(t)}[R]] = \mathbb{E}[v.\text{Conf}^{\,(t-1)}[R]] + \sum_{r=\alpha k}^{m} \Big(\sum_{j'=\alpha k}^{k} \frac{\binom{r-f}{j'}\binom{m-r+f}{k-j'}}{\binom{m}{k}} \Big) \binom{m}{r} p_i^r (1-p_i)^{m-r} \tag{6}$$

$$\mathbb{E}[v.\text{Conf}^{\,(t)}[B]] = \mathbb{E}[v.\text{Conf}^{\,(t-1)}[B]] + \sum_{r=\alpha k}^{m} \Big(\sum_{j'=\alpha k}^{k} \frac{\binom{r+f}{j'}\binom{m-r-f}{k-j'}}{\binom{m}{k}} \Big) \binom{m}{r} (1-p_i)^r {p_i}^{m-r}. \tag{7}$$

By defining $D \triangleq \Big(\mathbb{E}[v.\text{Conf}^{\,(t)}[B]] - \mathbb{E}[v.\text{Conf}^{\,(t-1)}[B]]\Big) - \Big(\mathbb{E}[v.\text{Conf}^{\,(t)}[R]] - \mathbb{E}[v.\text{Conf}^{\,(t-1)}[R]]\Big)$, we have

$$D = \sum_{r=\alpha k}^{m} \Big(\sum_{j'=\alpha k}^{k} \frac{\binom{r+f}{j'}\binom{m-r-f}{k-j'}}{\binom{m}{k}} \Big) \binom{m}{r} (1-p_i)^r {p_i}^{m-r} - \sum_{r=\alpha k}^{m} \Big(\sum_{j'=\alpha k}^{k} \frac{\binom{r-f}{j'}\binom{m-r+f}{k-j'}}{\binom{m}{k}} \Big) \binom{m}{r} p_i^r (1-p_i)^{m-r}. \tag{8}$$

We now aim to show that D acquires negative values, and once $v.\text{Conf}[B] - v.\text{Conf}[R]$ reaches value -1, all correct nodes acquire a red color. Let us introduce the following random variables $X_t \triangleq v.\text{Conf}^{\,(t)}[B] - v.\text{Conf}^{\,(t)}[R]$ and $X_{1:t} \triangleq \sum_{i=1}^{t} X_i$.

Since $X_{1:t}$ satisfies Hoeffding's inequality condition due to the fact that (*a*) X_t are i.i.d and (*b*) X_t's are sub-Gaussian due to their nature of having bounded values, we would have $P(X_{1:t} - \mathbb{E}[X_{1:t}] \geq q) \leq \exp(-2tq^2)$.

Therefore, in order to show $X_{1:t}$ is negative, with high probability, it suffices to show that $\mathbb{E}[X_{1:t}]$ is negative. To do so, based on the recursive formulas (6) and (7), we need to show that D acquires negative values under certain conditions. Let us first elaborate upon how (8) can be approximated as in the following Theorem.

Theorem 2. *D can be approximated as follows*

$$D \approx G(k, m, 1+\rho_b, \alpha, 1-p_i) - G(k, m, 1-\rho_b, \alpha, p_i), \tag{9}$$

where

$$G(k, m, \lambda, \alpha, p_i) \triangleq \sum_r \frac{1}{1 + e^{-1.702\left(\frac{\frac{k}{m}\lambda r - \alpha k}{\sqrt{\frac{k}{m}\lambda r(1-\frac{\lambda r}{m})}}\right)}} \frac{e^{-\frac{(r-mp_i)^2}{2\sigma_i^2}}}{\sqrt{2\pi\sigma_i^2}}, \tag{10}$$
$$\sigma_i^2 \triangleq m p_i (1-p_i).$$

Proof. We note that the probability mass function (PMF) of hyper-geometric distribution with parameters (r, m, k) is $p_{hg}(x; r, m, k) \triangleq \frac{\binom{r}{x}\binom{m-r}{k-x}}{\binom{m}{k}}$. By approximating

1. Binomial distribution *Bionomial*(n, p) (corresponds to terms $\binom{m}{r} p_i^r (1-p_i)^{m-r}$ or $\binom{m}{r} (1-p_i)^r {p_i}^{m-r}$) with normal distribution $\mathcal{N}(np, np(1-p))$,

2. Hyper-geometric distribution $Hyper - Geometrical(r, m, k)$ with normal distribution $\mathcal{N}(k\frac{r}{m}, k\frac{r}{m}(1 - \frac{r}{m}))$,
   D can be approximated as

$$\begin{aligned} D \approx \sum_{r} \Big(1 - \Phi(\frac{\alpha k - k\frac{r+f}{m}}{\sqrt{k\frac{r+f}{m}(1 - \frac{r+f}{m})}})\Big) \frac{e^{-\frac{(r-m(1-p_i))^2}{2\sigma_i^2}}}{\sqrt{2\pi\sigma_i^2}} \\ - \sum_{r} \Big(1 - \Phi(\frac{\alpha k - k\frac{r-f}{m}}{\sqrt{k\frac{r-f}{m}(1 - \frac{r-f}{m})}})\Big) \frac{e^{-\frac{(r-mp_i)^2}{2\sigma_i^2}}}{\sqrt{2\pi\sigma_i^2}}. \end{aligned} \tag{11}$$

where $\Phi(x)$ represents the cumulative distribution function (CDF) of normal distribution $\mathcal{N}(0, 1)$. According to [33], $\Phi(x) \approx \frac{1}{1+e^{-1.702x}}$. Therefore, by substituting $f = \rho_b r$ and using the aforementioned approximation of $\Phi(x)$, we can approximate D as (9). □

Remark 1. *One can easily see that D can acquire negative values when $p_i > \frac{1}{2}$. This is due to the fact that the logistic coefficient of the normal distribution in $G(.)$ considerably scales down normal distribution $\mathcal{N}(m(1 - p_i), mp_i(1 - p_i))$ (appeared in $G(k, m, 1 + \rho_b, \alpha, 1 - p_i)$) compared to normal distribution $\mathcal{N}(mp_i, mp_i(1 - p_i))$ (appearing in $G(k, m, 1 - \rho_b, \alpha, p_i)$).*

Remark 2. *As k increases, the logistic term in $G(.)$ forms a sharper transition around the central point (value that makes logistic term equal to $\frac{1}{2}$). Therefore, for a sufficiently large k, D would be negative if $\frac{m\alpha}{1+\rho_b} > m(1 - p_i)$ and $\frac{m\alpha}{1-\rho_b} < mp_i$, due to the fact that the mean of $\mathcal{N}(m(1 - p_i), mp_i(1 - p_i)) <$ central point of logistic coefficient of $\mathcal{N}(m(1 - p_i), mp_i(1 - p_i))$ and the mean of $\mathcal{N}(mp_i, mp_i(1 - p_i)) >$ central point of logistic coefficient of $\mathcal{N}(mp_i, mp_i(1 - p_i))$. The aforementioned conditions are equivalent to $p_i > \max(\frac{\alpha}{1-\rho_b}, 1 - \frac{\alpha}{1+\rho_b})$.*

Determining Δ: The proper choice of i can be obtained by finding the least integer i, where D is negative. By denoting the appropriate i as i^*, then Δ (presented in Step 1) can be found by solving $\frac{c}{2} + \Delta = i^*$.

All tuples (ρ_n, ρ_b) for which the safety is guaranteed can be obtained by checking if there exists an i such that $D < 0$ for those values of ρ_n and ρ_b.

We further perform simulations to obtain all tuples (ρ_n, ρ_b) such that safety is assured for the case of having 2000 mobile devices. Figure 3 illustrates this guaranteed safety region, shown with yellow color, for different numbers of brokers and different numbers of connections. We consider 1000 iterations with a 0.05 resolution for the Byzantine ratio of nodes and brokers. One interesting observation is that the Byzantine ratio of mobile nodes and brokers can, respectively, reach 50% (when ρ_b is small) and <60% of Byzantine brokers (when ρ_n is small), with a tradeoff seen between these ratios. Notably, if the ratio of Byzantine nodes increases, the maximum tolerable ratio of Byzantine brokers for ensuring safety decreases, and vice versa. This dynamic is particularly relevant when considering the specific behaviors of Byzantine entities in Blizzard. Malicious brokers are mostly limited to suppressing messages, as they cannot initiate or forge messages, while Byzantine mobile nodes, although computationally constrained and unable to forge signatures, can adopt any execution strategy. It is crucial to recognize that mobile devices are inherently more susceptible to failures, such as battery drain or network disconnections, which, within the context of the Blizzard protocol, are treated akin to malicious behaviors. However, the protocol is designed to operate effectively as long as the combined proportion of Byzantine mobile nodes and brokers falls within the safety-guaranteed region. Moreover, brokers, due to their typically more stable infrastructure, are less prone to the types of failures common to mobile devices, allowing for a higher tolerable ratio of Byzantine brokers compared to mobile nodes. This distinction underscores the strategic advantage of investing in the integrity and reliability of brokers. By ensuring a higher proportion of correct brokers, Blizzard can maintain operational safety even with a greater ratio of mobile nodes exhibiting malicious or failure-induced behaviors. This insight highlights the protocol's capacity to adapt to and mitigate the risks associated with the inherent vulnerabilities of mobile nodes, further enhancing Blizzard's resilience and operational efficacy in environments with diverse Byzantine threats. One notable observation is that the safety region expands as k increases. This phenomenon can be attributed to the intuitive understanding that a higher value of k indicates a greater involvement of nodes in the transaction query.

The safety region plot effectively showcases the resilience of the Blizzard protocol against adversarial behaviors, particularly focusing on the range of actions that Byzantine nodes and brokers might undertake. It is crucial to note that while correct mobile nodes consistently respond truthfully, Byzantine nodes may offer any response. Additionally, while correct brokers broadcast the η-majority vote, Byzantine brokers, despite their inability to forge information, can selectively compute and communicate the η-majority from a subset of nodes. This nuanced behavior underlines the importance of the safety region in ensuring that all correct nodes decide on the same transaction among conflicting ones, a vital aspect for maintaining integrity in real-world scenarios where diverse adversarial tactics may be employed. For example, if there are 170 brokers in the network and around 20% of them are malicious, then according to the safety-guaranteed region in Figure 3, the Blizzard protocol guarantees reaching consensus as long as the portion of Byzantine mobile nodes does not exceed 40% of all mobile nodes. This demonstrates Blizzard's capacity to uphold consensus and safety even in significantly adverse conditions, emphasizing its practical utility in distributed systems where the presence of malicious actors is a real concern.

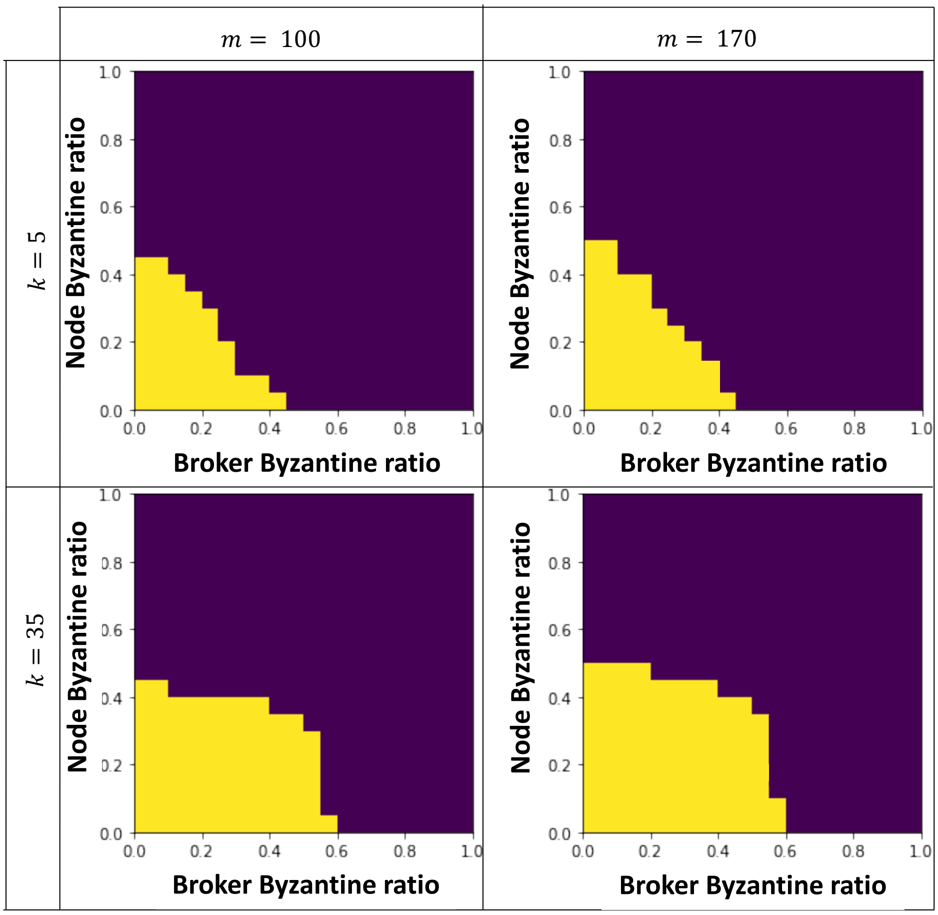

Figure 3. An illustration of safety-guaranteed region (indicated with yellow color) of Blizzard protocol through simulation for different number of connections of each mobile node, i.e., k, and different number of brokers, i.e., m while fixing total number of mobile nodes $n = 2000$ and 1000 iterations.

4.2. Liveness

Similar to other DAG-based protocols such as IOTA [18] and Avalanche [15], liveness failure in Blizzard occurs when either a transaction has an invalid transaction as its parent or a transaction does not gain enough confidence value. The former scenario can be resolved by re-issuing the transaction with new valid parents, while the latter could be resolved by having a node send additional valid transactions as successors to increase the confidence value.

5. Performance Analysis

We first present our analysis on throughput per shard, and then we focus on analyzing latency.

5.1. Throughput per Shard

We elaborate upon obtaining the throughput of Blizzard from a novel perspective, i.e., modeling it as a pipeline, as illustrated in Figure 4. By considering t_i as the required time for performing the task of component i, and considering that the component with the smallest rate would dominate the result, the throughput would be equal to $\min_{1 \le i \le 8} \frac{1}{t_i}$.

Considering the network bandwidth as BW bps and the transaction size as 300 Bytes, the rate of the communication components (represented by green-colored boxes, specifically Components 2–3, 5, and 7) would be approximately $\frac{BW}{2400}$ transactions per second (tps). Among the computing components (orange-colored boxes), i.e., Components 1, 4, 6, and 8, Component 4 (checking if a transaction is strongly preferred) dominates the computing time due to the fact that it is more time-consuming compared to the other computing components. Using this analysis, we will quantify the throughput using experimental measurements in Section 6.

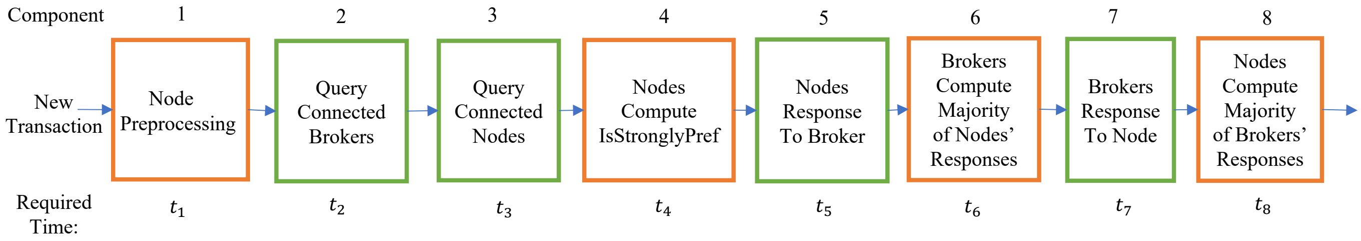

Figure 4. The pipeline modeling of different components of Blizzard to acquire throughput. We use orange for computing components and green for communication components in our box diagrams.

5.2. Latency

Total latency refers to the time interval from when a transaction is issued until it is finalized. It is upper-bounded by the sum of three terms:

$$\text{Latency} \le t_{\text{propagation}} + t_{\text{validation}} + t_{\text{confidence}} \tag{12}$$

These terms are, respectively, (1) the *propagation time* $t_{\text{propagation}}$, which is the time taken for a transaction to be disseminated throughout the entire network; (2) the *transaction validation time* $t_{\text{validation}}$; and (3) the *confidence-gathering time* $t_{\text{confidence}}$, which is the time required for a transaction to achieve a threshold level of confidence through successive transactions voting for it. We analyze each of the aforementioned terms as follows.

5.2.1. Propagation Time

Since the propagation time is linearly proportional to the number of communication rounds (a communication round is a communication transmission from a mobile node to a broker or vice versa), we aim to obtain the least number of communication rounds (LNCR) required for a transaction to propagate in the entire network by starting from the node that issued this transaction and ending by the last node which discovers this transaction. We are able to prove a strong result about LNCR in Blizzard.

Theorem 3. *Let us assume that* $\frac{(m-k)n}{(m-1)m} > 1$*; then, the LNCR of Blizzard equals 4 with high probability.*

Proof. We first demonstrate that, with high probability, the distance between any two vertices representing brokers in the corresponding bipartite graph is 2. Then, the distance between any two vertices indicating mobile nodes would be at most 2 more than that, i.e., 4 with high probability. The probability that vertices v_1 and v_2, $\forall v_1 \neq v_2$ and $v_1, v_2 \in V$, have a distance of 2, which can be obtained as follows:

$$\begin{aligned} P(dist(v_1, v_2) = 2) &= P(\text{existence of at least one node} \\ &\quad u \in U \text{ which is connected to both } v_1 \text{ and } v_2) \\ &\overset{(a)}{=} 1 - \left(1 - \frac{\binom{m-2}{k-1}}{\binom{m-1}{k-1}}\right)^{\frac{n}{m}} = 1 - \left(1 - \frac{m-k}{m-1}\right)^{\frac{n}{m}} \overset{(b)}{\approx} 1 - e^{-r} \end{aligned}$$

where (a) follows from considering average $\frac{n}{m}$ mobile nodes per broker, and (b) follows from $\lim_{n\to\infty}(1 - \frac{r}{n})^n = e^{-r}$ and considering $r := \frac{(m-k)n}{(m-1)m}$. It is easy to see e^{-r} is small enough if $r > 1$ (or equivalently the condition of the Theorem holds). This condition is realistic due to the fact that we expect $n >> m$. □

5.2.2. Transaction Validation Time

As mentioned earlier, the transaction validation time refers to the time required for a node to check the validity of transactions. Therefore, we can use the pipeline scheme, explained in the previous section on throughput and illustrated in Figure 4, to model this time. Mathematically, the transaction validation time can be expressed as $\sum_{i=1}^{8} t_i$.

5.2.3. Confidence-Gathering Time

Let us assume L represents the average number of transactions come later after a transaction appended to the DAG until it gets finalized. The value of L would depend on the security parameters of the protocol as well as the DAG-attachment policy adopted by nodes; in the special case when all transactions are attached sequentially in a chain, it can be shown that L would be equal to $\min(\beta_1, \beta_2)$. By defining ζ as the arrival rate of transactions, the average confidence-gathering time would be $\frac{L}{\zeta}$.

Using the above analysis, we will quantify the latency using experimental measurements in Section 6.

5.3. Average Message Complexity

Average message complexity refers to the average number of messages required for a transaction to be queried by all nodes. The average message complexity of Blizzard can be obtained by noting that each broker is queried by one of its connected nodes and then collects and sends back the majority vote to all its connected nodes. This implies that Blizzard needs $m + 2kn$ messages for querying.

While we argue that Avalanche cannot be implemented on mobile device networks in a scalable manner as it requires direct peer-to-peer communication between any two random devices, we can still compare Blizzard and Avalanche in terms of their trade-off between message complexity and LNCR, assuming Blizzard were to be implemented on the same network as Avalanche, as shown in Figure 5. For having a fair comparison (i.e., equal number of nodes being queried on each transaction), q (number of sampled nodes in Avalanche) should be $\frac{nk}{m}$. As q increases in the Avalanche protocol, the LNCR decreases, while the total number of required messages significantly increases. However, Blizzard protocol obtains the best of both worlds, i.e., having a lower LNCR and lower total number of required messages. Note that the number of required messages in Blizzard could be reduced further by decreasing m, but this would result in lower security.

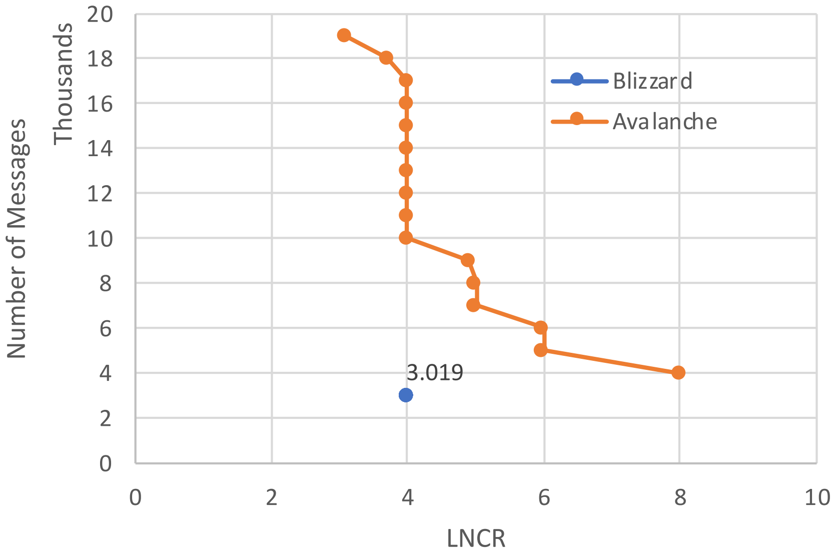

Figure 5. The total number of massages versus the lowest number of communication rounds (LNCR) in Blizzard compared with Avalanche for the following setting: $n = 500$, $k = 3$, and m is varied from 4 to 19.

6. Implementation and Experimental Measurements

We implemented Blizzard in C++ and also a version of Avalanche for comparison purposes. We have made our source code for the implementation of Blizzard as well as its comparison with Avalanche publicly available at https://github.com/ANRGUSC/Blizzard (accessed on 15 December 2022).

Using our implementation, we ran computations of Blizzard on a computing machine with the configuration of 2.3 GHz Intel core i5 (whose frequency is less than the frequency of state-of-the-art mobile CPUs [34]) so that all computations are emulated in real time. For the communication between nodes and brokers, we simulated it with one-way network latency drawn from a uniform distribution with 100 ms mean and standard deviation 25 ms. We next present the experimental measurements of throughput and latency, as well as an estimation of battery energy consumption.

6.1. Throughput Evaluation

As we explained in Section 5.1, since Component 4 (checking if a transaction is strongly preferred) in Figure 4 dominates the computational time, we implemented component 4 in C++ and observed empirically that $t_4 \approx 100$ µs for the setting where there are 400 transactions known by mobile nodes. Since the computing power of the top 10 iOS mobile devices exceeds the computing power of the device used in the implementation we performed (based on https://www.geekbench.com/ (accessed on 15 November 2023)), the throughput of Blizzard is as presented in the following table:

Network Bandwidth	100 Mbps	10 Mbps	1 Mbps
Throughput on PCs	10,000 TPS	4166 TPS	416 TPS

6.2. Latency Evaluation

Assuming security parameters $\beta_1 = 11$, $\beta_2 = 150$, a chain topology for the DAG and a transaction arrival rate higher than 100 tps, the propagation and validation times are going to be dominant compared to the confidence time, and in turn, they will each be dominated by four communication steps; this implies a total latency on the order of <1 s (∼0.65 s).

Figure 6 shows the histograms of the transaction latency of our proposed scheme and Avalanche [15] in two different settings. As it can be observed, Blizzard significantly reduces latency by ∼50%. Further, one can see Avalanche has a wide range of transaction latency while Blizzard has a dense one.

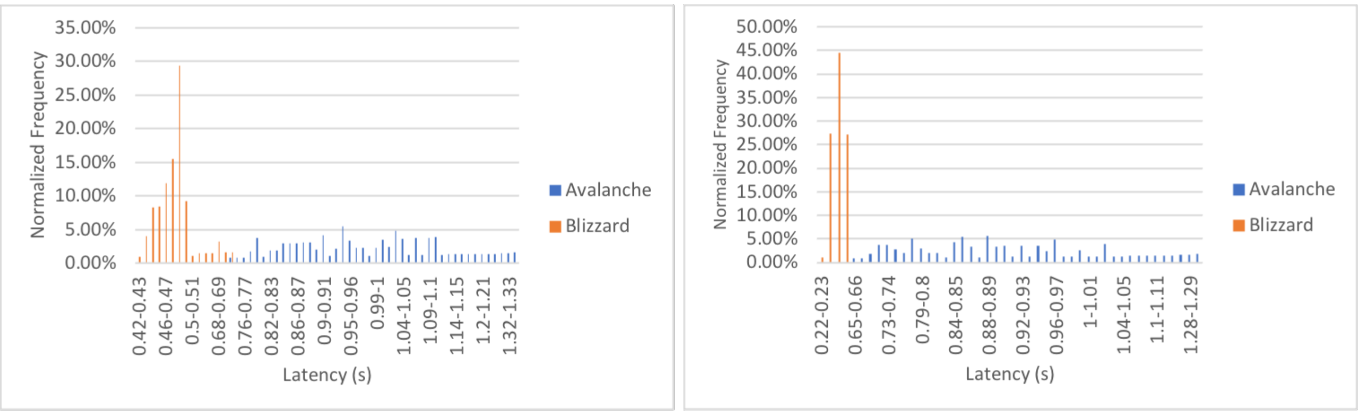

Figure 6. The **left** and **right** plots, respectively, represent the histogram of transaction latency of our proposed scheme (Blizzard) compared with Avalanche [15] for the case of 100 transactions, security parameters $\beta_1 = 11$ and $\beta_2 = 150$. The **left** histogram corresponds to the settings of 100 nodes, 8 brokers, and 3 connections per node, while the **right** figure corresponds to the case of 200 nodes, 11 brokers, and 6 connections per node.

6.3. Battery Energy Consumption

Here, we provide a rough estimate of the energy consumed per transaction validated by a mobile device. Given that the amount of data exchanged between the mobile nodes and online brokers for each transaction is relatively small, and considering that no computationally intensive Sybil control mechanism, such as proof of work, is required, the primary source of energy consumption in our protocol is computation. As discussed above, the dominant source of computation when validating a transaction (see Figure 4) is Component 4 (Computing IsStronglyPref.). Based on our experiments, we estimate this takes about 100 µs. Assuming a mobile CPU power consumption of 1.5 Watt [34] and conservatively assuming full CPU utilization, this would translate to about 1.5×10^{-4} Joules per transaction. While we are not aware of benchmark numbers for other protocols we could compare this with since other protocols are typically not built with the energy efficiency of validators in mind, we believe this imposes a relatively low, manageable load on a mobile device, particularly as the device owner is free to determine what number of transactions it participates in over a given period of time.

To succinctly highlight the differences between Blizzard and Avalanche, we present the comparative Table 3.

Table 3. Comparison of different protocols.

Protocol	Transaction per Second per Shard	Confirmation Latency	Number of Validators	Mobile-Based
Avalanche [15]	∼3400	∼1.35 s	100k+	No
Blizzard	**∼10,000**	**∼0.65 s**	**100M+**	**Yes**

7. Discussion

Here, we briefly elaborate on topics that merit further attention. These topics are out of scope for the present paper, but we are actively pursuing these directions.

7.1. Mobile-Device-Oriented Sybil Control

What has been presented in this paper thus far is a consensus mechanism. It implicitly assumes that there is already a Sybil control mechanism in place, such as those based on proof of work or proof of stake. To implement Blizzard in a network with millions of mobile devices, it may be helpful to create a Sybil control mechanism such that only users with valid mobile devices can participate in the consensus with significant cost associated with creating or operating multiple, potentially fake identities. A potential design for such a system could leverage the existence of globally unique mobile IDs, such as IMEI numbers, while still maintaining an overall architecture that is sufficiently open and decentralized. Another alternative is to utilize decentralized IDs on the mobile nodes with a permissioned setup. Yet another approach may be to use location information or wireless signal strength as the basis for a Sybil control mechanism [35,36]. Another approach could be to use proof of social Contacts [37], which leverages encounter information between mobile nodes to detect and blacklist Sybil nodes.

7.2. Improving Scalability

One of the crucial bottlenecks of a DAG-based protocol is that all nodes need to store the entire DAG as well as investigate whether a transaction is strongly preferred by going through the entire DAG. As a result, the system faces storage and computation bottlenecks. These challenges may be exacerbated when involving relatively more resource-limited mobile nodes in the consensus mechanism, as we have proposed in Blizzard. To address the computation and storage challenge, we consider three approaches, namely *sharding* [38], *pruning DAG* , and *offloading verification*, to improve the scalability.

Sharding: This technique creates multiple pools of mobile nodes, where each pool focuses on storing and verifying transactions belonging to a corresponding subset of all accounts (per the well-known Blockchain Trilemma, this solution trades off security for scalability while maintaining decentralization). The implementation of sharding is particularly beneficial in mobile environments as it helps alleviate computational and storage constraints on cell phones, allowing them to participate more effectively in the consensus process without being overwhelmed by the demands of the entire network.

Pruning DAG: By defining *check-point transaction* as the one that is finalized, it is clear that all ancestor transactions of a check-point transaction are also finalized. In the account model (the state-based approach used in Ethereum), as long as we reach a check-point transaction on the DAG, we do not need to store all its ancestor transactions. Therefore, each mobile node can save a significant amount of memory by storing only the pruned DAG. By including such a concept of check points, we could fulfill the criterion of having lightweight nodes. However, as discussed in [39], there are several other practical considerations to keep in mind, such as retaining historical information on some full nodes and ensuring it is accessible in a decentralized manner for security purposes. In our architecture, the online brokers could potentially serve the role of full nodes that store the entire history, while the mobile nodes only store information past the last check-point. This approach significantly assists in easing the memory storage constraints on mobile devices, enabling them to participate in the network without the burden of storing extensive historical data.

Off–loading Verification: Inspired by [40], this approach would allow the computation associated with verification (investigating whether a transaction is strongly preferred or not) to be offloaded to more powerful servers that provide zero-knowledge proofs that can be verified in a more lightweight manner by the mobile devices. More research is needed to flesh out and realize such an approach. This method is particularly advantageous for mobile devices, as it addresses the computation power constraint, enabling them to efficiently verify transactions without expending significant processing resources.

7.3. Safety under a Partially Synchronous Model

As shown in [31], in order to prove safety under a partially synchronous model, the protocol must explicitly take into account the total number of participating nodes and their votes when determining when to finalize a transaction, as seen in protocols such as Tendermint [11] and Hotstuff [12]. Alternatively, it would be interesting to explore the development of a decentralized BFT approach to empirically quantifying (possibly in a time-varying manner) an upper bound on the network latency at all times. Given such a mechanism, the protocol parameters could be suitably adapted to ensure correct and efficient operation despite a time-varying network latency, i.e., in a partially synchronous network.

7.4. Connectivity

Blizzard is designed to function in the unpredictable and unreliable network environments characteristic of mobile devices. If a mobile node disconnects, the integrity of the system is maintained as long as the proportion of malicious entities is within the specified safe range. The only requirement for the disconnected node upon reconnection is to update its DAG ledger, which is efficiently achieved through communication with the connected brokers.

In our protocol, safety is effectively ensured through the strategy of establishing random connections to brokers. This approach is critical in protecting against coordinated adversarial attacks. To further adapt our protocol to the dynamic nature of user mobility, we propose leveraging advancements in telecommunications like 5G and the forthcoming 6G networks. These technologies will enable a more densely connected network of brokers, enhancing our system's resilience to both adversarial threats and the complexities introduced by mobile user behavior. This strategy of combining random broker connections with increased broker density is key to evolving our protocol for more realistic network scenarios.

7.5. Challenges in Real-World Implementation

There are several key challenges pertinent to the implementation of consensus protocols on mobile devices. One significant hurdle is the resource constraint inherent in mobile devices, which possess limited battery life compared to traditional computing nodes. Consequently, optimizing consensus protocols for low resource consumption is critical to prevent excessive battery drain and performance degradation. Moreover, from an implementation standpoint, it is imperative that every mobile node and broker has access to the distributed random beacon. This access is vital for facilitating random connections between mobile devices and brokers, leading to additional considerations regarding the network's architecture, security, and dependability. The challenge of providing stable and secure beacon access in a broad and fluctuating network of mobile devices is unique and must be tackled to preserve the consensus protocol's integrity and operational effectiveness. Another crucial aspect to consider is the impact of these protocols on user experience. It is imperative that the running of consensus protocols in the background does not intrusively affect the primary functionalities of the device, ensuring that they operate seamlessly without significantly impacting the user's experience with the device.

8. Conclusions

In this work, we have presented Blizzard, the first mobile-device-oriented BFT consensus-based distributed ledger protocol. Blizzard incorporates a novel two-tier broker-based architecture and a decentralized random matching mechanism. We have mathematically analyzed and presented the safety guarantee for Blizzard. Interestingly this guarantee is in the form of a two-dimensional region—for sufficiently large networks, our numerical computations show that the protocol is capable of supporting $< 50\%$ of Byzantine nodes (when the number of Byzantine brokers is small) and $<60\%$ of Byzantine brokers (when the number of Byzantine nodes is small), with a tradeoff seen between these ratios.

We have also discussed how Blizzard satisfies liveness. Moreover, we have also analyzed and evaluated the performance of Blizzard in terms of throughput, latency, and message complexity. We also demonstrated that Blizzard has superior performance in terms of significantly low message complexity and short propagation latency compared to its benchmark, which in any case would be challenging to implement in a mobile-first scenario considered here as it is challenging to allow mobile devices to directly randomly query any other mobile nodes in a large network without going through any online servers.

We have shown that Blizzard can provide a desirable level of throughput per shard. To improve throughput performance further, it is important to develop or adopt additional scaling mechanisms that have been proposed in other projects, such as account-based sharding and second-layer solutions such as state channels. Memory limitations of mobile devices can be addressed by a suitable combination of check-point-based DAG pruning and prior offloaded verification solutions. As a key future direction, we aim to implement Blizzard on real mobile devices and empirically measure the throughput and latency.

Author Contributions: Conceptualization, M.K., B.K. and S.Y.; Data curation, M.K. and B.K.; Formal analysis, M.K. and B.K.; Funding acquisition, B.K. and S.Y.; Investigation, M.K., B.K. and S.Y.; Methodology, M.K. and B.K.; Project administration, B.K. and S.Y.; Resources, M.K., B.K. and S.Y.; Software, M.K. and B.K.; Supervision, B.K. and S.Y.; Validation, M.K., B.K. and S.Y.; Visualization, M.K. and B.K.; Writing—original draft, M.K. and B.K.; Writing—review and editing, M.K., B.K. and S.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by a gift to the USC Center for Cyberphysical Systems and the Internet of Things from SovereignWallet Network and in part by AFOSR grant FA9550-23-1-0312. We also acknowledge the helpful inputs provided by Muhammad Naveed.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/ANRGUSC/Blizzard (accessed on 13 February 2024).

Conflicts of Interest: Seokgu Yun is employed by the SovereignWallet Network. Other authors declare no conflict of interest.

Appendix A. High-Probability Connections

By defining random variable X as a number of Byzantine brokers picked by selecting ℓ brokers from all brokers, $P(X = x) = \frac{\binom{m_b}{x}\binom{m_c}{\ell-x}}{\binom{m_b+m_c}{\ell}}$. Based on the Hoeffding inequality, we have $P(X - \mathbb{E}[X] > \theta) \leq e^{-\frac{2\theta^2}{\ell^2}}$, where $\mathbb{E}[X] = \frac{m_b}{m}\ell = \rho_b \ell$.

Appendix B. Overview on DAG Structure of the Ledger

A mobile node designates several parents for a new transaction once upon issuing a new transaction and forms edges on the DAG. The main difficulty of keeping the DAG is to select one transaction among conflicting ones. Double-spending is one of the examples of conflicting transactions. Once a transaction is queried, all ancestor transactions of this one are implicitly included in the query. A node affirmatively responds to the query if all of the ancestors are currently the *preferred* choice of transaction in their corresponding conflict sets. In the case of having $\geq \alpha k$ (where $\frac{1}{2} < \alpha < 1$) positive response for transaction T, then this transaction receives voucher $v_{u,T} = 1$ and be appended to the DAG; otherwise $v_{u,T} = 0$.

Every node stores the entire transactions it has known in its DAG. Each DAG consists of mutually exclusive conflict sets $\mathcal{P}_T$ for $T \in \mathcal{T}_u$, where $\mathcal{T}_u$ represents the subset of the known transaction by node u. Each conflict set $\mathcal{P}_T$ has three components, namely the preferred transaction $\mathcal{P}_T$.pref, last seen transaction $\mathcal{P}_T$.last, and counter $\mathcal{P}_T$.counter. Moreover, every node u computes the *confidence* value of transaction T by the following formula:

$$d_u(T) \triangleq \sum_{T' : T' \in \mathcal{T}_u, T' \rightsquigarrow T} v_{u,T'} \tag{A1}$$

where $T' \rightsquigarrow T$ indicates a path from T' to T. Furthermore, DAGs created by different nodes are assured to be consistent, meaning that relation $T \to T'$ exists for the DAG of all nodes if $T \to T'$, and there is no node with relation $T \to T'$ if $T \not\to T'$. Each node maintains a counter for the number of votes it has received for each transaction. Once the number of affirmative votes surpasses a certain predefined threshold, the transaction is considered finalized.

References

1. Walden, P.; Dahlberg, T.; Penttinen, E. Introduction to the Minitrack on Digital Mobile Services for Everyday Life. In Proceedings of the 52nd Hawaii International Conference on System Sciences, Maui, HI, USA, 8–11 January 2019.
2. Greenstein, B. Delivering the Mobile Web to the Next Billion Users. In Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications, Tempe, AZ, USA, 12–13 February 2018; p. 99.
3. Biryukov, A.; Tikhomirov, S. Security and privacy of mobile wallet users in Bitcoin, Dash, Monero, and Zcash. *Pervasive Mob. Comput.* **2019**, *59*, 101030. [CrossRef]
4. Sai, A.R.; Buckley, J.; Le Gear, A. Privacy and Security Analysis of Cryptocurrency Mobile Applications. In Proceedings of the 2019 Fifth Conference on Mobile and Secure Services (MobiSecServ), Miami, FL, USA, 2–3 March 2019; pp. 1–6.
5. Yu, K.; Feng, J. Moore's Law and Price Trends of Digital Products: The Case of Smartphones. *Econ. Innov. New Technol.* **2019**, *10*, 1628509.
6. Kiamari, M.; Krishnamachari, B.; Naveed, M.; Yun, S. Distributed Consensus for Mobile Devices using Online Brokers. In Proceedings of the 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Toronto, ON, Canada, 2–6 May 2020; pp. 1–3.
7. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 15 December 2019).
8. Garay, J.; Kiayias, A.; Leonardos, N. The bitcoin backbone protocol: Analysis and applications. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, 26–30 April 2015.
9. Castro, M.; Liskov, B. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst. (TOCS)* **2002**, *20*, 398–461. [CrossRef]
10. Bessani, A.; Sousa, J.; Alchieri, E.E. State machine replication for the masses with BFT-SMaRt. In Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Atlanta, GA, USA, 23–26 June 2014; pp. 355–362.
11. Kwon, J. Tendermint: Consensus without Mining. 2014. Available online: http://tendermint.com/docs/tendermint.pdf (accessed on 15 November 2023).
12. Abraham, I.; Gueta, G.; Malkhi, D. Hot-Stuff the Linear, Optimal-Resilience, One-Message BFT Devil. *CoRR* **2018**, abs/1803.05069.
13. Zamfir, V.; Rush, N.; Asgaonkar, A.; Piliouras, G. *Introducing the "Minimal CBC Casper" Family of Consensus Protocols*; DRAFT v1. 0; Ethereum Research: Zug, Switzerland, 2018.
14. Baird, L. *The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance*; Swirlds Tech Report SWIRLDS-TR-2016-01; Swirlds Inc.: College Station, TX, USA, 2016; Volume 34, pp. 9–11.
15. Rocket, T.; Yin, M.; Sekniqi, K.; van Renesse, R.; Sirer, E.G. Scalable and Probabilistic Leaderless BFT Consensus through Metastability. *arXiv* **2019**, arXiv:1906.08936.
16. Crain, T.; Gramoli, V.; Larrea, M.; Raynal, M. DBFT: Efficient Leaderless Byzantine Consensus and its Application to Blockchains. In Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 1–3 November 2018; pp. 1–8.
17. Gagol, A.; Swietek, M. Aleph: A Leaderless, Asynchronous, Byzantine Fault Tolerant Consensus Protocol. *arXiv* **2018**, arXiv:1810.05256.
18. Popov, S. The Tangle. 2016. Available online: https://www.iota.org/research/academic-papers (accessed on 15 December 2019).
19. Zhelezov, D.; Fohrmann, O. HelixMesh: A Consensus Protocol for IoT. In Proceedings of the 2019 International Electronics Communication Conference, Okinawa, Japan, 7–9 July 2019; pp. 44–51.
20. Cosmos. Available online: https://cosmos.network/resources/whitepaper (accessed on 18 December 2019).
21. Micali, S. ALGORAND: The Efficient and Democratic Ledger. *arXiv* **2016**, arXiv:1607.01341.
22. Kiayias, A.; Russell, A.; David, B.; Oliynykov, R. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. Cryptology ePrint Archive, Report 2016/889. 2016. Available online: https://eprint.iacr.org/2016/889 (accessed on 15 November 2023).
23. Hanke, T.; Movahedi, M.; Williams, D. DFINITY Technology Overview Series, Consensus System. *CoRR* **2018**, abs/1805.04548.
24. Fairley, P. Ethereum will cut back its absurd energy use. *IEEE Spectr.* **2018**, *56*, 29–32. [CrossRef]
25. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. In Proceedings of the Thirteenth EuroSys Conference, Porto, Portugal, 23–26 April 2018.
26. Olson, K.; Bowman, M.; Mitchell, J.; Amundson, S.; Middleton, D.; Montgomery, C. *Sawtooth: An Introduction*; The Linux Foundation: San Francisco, CA, USA, 2018.
27. Wu, W.; Cao, J.; Yang, J.; Raynal, M. A hierarchical consensus protocol for mobile ad hoc networks. In Proceedings of the 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP'06), Sochaux, France, 15–17 February 2006; p. 9. [CrossRef]

28. Badache, N.; Hurfin, M.; Macedo, R. Solving the consensus problem in a mobile environment. In Proceedings of the 1999 IEEE International Performance, Computing and Communications Conference (Cat. No.99CH36305), Scottsdale, AZ, USA, 10–12 February 1999; pp. 29–35.
29. Wu, W.; Cao, J.; Raynal, M. Eventual Clusterer: A Modular Approach to Designing Hierarchical Consensus Protocols in MANETs. *IEEE Trans. Parallel Distrib. Syst.* **2009**, *20*, 753–765. [CrossRef]
30. Wu, W.; Cao, J.; Yang, J.; Raynal, M. Design and Performance Evaluation of Efficient Consensus Protocols for Mobile Ad Hoc Networks. *IEEE Trans. Comput.* **2007**, *56*, 1055–1070. [CrossRef]
31. Pass, R.; Shi, E. Hybrid Consensus: Efficient Consensus in the Permissionless Model. In Proceedings of the 31st International Symposium on Distributed Computing (DISC 2017), Vienna, Austria, 16–20 October 2017; pp. 1–16.
32. Syta, E.; Jovanovic, P.; Kogias, E.K.; Gailly, N.; Gasser, L.; Khoffi, I.; Fischer, M.J.; Ford, B. Scalable Bias-Resistant Distributed Randomness. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–24 May 2017; pp. 444–460.
33. Bowling, S.; Khasawneh, M.; Kaewkuekool, S.; Cho, B. A logistic approximation to the cumulative normal distribution. *J. Ind. Eng. Manag.* **2009**, *2*, 114–127. [CrossRef]
34. Halpern, M.; Zhu, Y.; Reddi, V.J. Mobile CPU's rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction. In Proceedings of the 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), Barcelona, Spain, 12–16 March 2016; pp. 64–76.
35. Jiang, Z.; Krishnamachari, B.; Zhou, S.; Niu, Z. SENATE: A Permissionless Byzantine Consensus Protocol in Wireless Networks. *arXiv* **2018**, arXiv:1803.08694.
36. King, R.J. Introduction to Proof of Location: The Case for Alternative Location Systems. 2018. FOAM. Available online: https://blog.foam.space/introduction-to-proof-of-location-6b4c77928022 (accessed on 15 November 2023).
37. Martinez, M.; Hekmati, A.; Krishnamachari, B.; Yun, S. Mobile Encounter-based Social Sybil Control. In Proceedings of the 2nd International Workshop on Blockchain Applications and Theory (BAT), Paris, France, 20–23 April 2020.
38. On Sharding Blockchains, Ethereum Wiki. Available online: http://github.com/ethereum/wiki/wiki/Sharding-FAQ (accessed on 15 November 2023).
39. On Pruning in Ethereum. Available online: http://tiny.cc/ethpruning (accessed on 15 November 2023).
40. Al-Bassam, M.; Sonnino, A.; Buterin, V. Fraud Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities. *arXiv* **2018**, arXiv:1809.09044.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.