Practical Provably Secure Flooding for Blockchains

Chen-Da Liu-Zhang*[1], Christian Matt[2], Ueli Maurer[3],
Guilherme Rito[3], and Søren Eller Thomsen[4]

[1]Carnegie Mellon University, Pittsburgh, USA
cliuzhan@andrew.cmu.edu
[2]Concordium, Zurich, Switzerland
cm@concordium.com
[3]Department of Computer Science, ETH Zurich, Switzerland
{maurer, gteixeir}@inf.ethz.ch
[4]Concordium Blockchain Research Center, Aarhus University, Denmark
sethomsen@cs.au.dk

May 17, 2022

Abstract

In recent years, permisionless blockchains have received a lot of attention both from industry and academia, where substantial effort has been spent to develop consensus protocols that are secure under the assumption that less than half (or a third) of a given resource (e.g., stake or computing power) is controlled by corrupted parties. The security proofs of these consensus protocols usually assume the availability of a network functionality guaranteeing that a block sent by an honest party is received by all honest parties within some bounded time. To obtain an overall protocol that is secure under the same corruption assumption, it is therefore necessary to combine the consensus protocol with a network protocol that achieves this property under that assumption. In practice, however, the underlying network is typically implemented by flooding protocols that are not proven to be secure in the setting where a fraction of the considered total weight can be corrupted. This has led to many so-called eclipse attacks on existing protocols and tailor-made fixes against specific attacks.

To close this apparent gap, we propose a flooding protocol that provably delivers sent messages to all honest parties after a logarithmic number of steps. We prove security in the setting where all parties are publicly assigned a positive weight and the adversary can corrupt parties accumulating up to a constant fraction of the total weight. This can directly be used in the proof-of-stake setting, but is not limited to it. To prove the security of our protocol, we combine known results about the diameter of Erdős–Rényi graphs with reductions between different types of random graphs. We further show that the efficiency of our protocol is asymptotically optimal.

The practicality of our protocol is supported by extensive simulations for different numbers of parties, weight distributions, and corruption strategies. The simulations confirm our theoretical results and show that messages are delivered quickly regardless of the weight distribution, whereas protocols that are oblivious of the parties' weights completely fail if the weights are unevenly distributed. Furthermore, the average message complexity per party of our protocol is within a small constant factor of such a protocol. Hence, security in a weighted setting essentially comes for free with our techniques.

*Supported in part by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

Contents

1 Introduction

Since Nakamoto proposed the first decentralized permisionless blockchain protocol [Nak08], a significant line of works has been done. In such protocols, one considers a setting where different parties are weighted according to how much amount of a resource they own (mining power, stake, space, etc.), and security relies on the fact that a certain fraction of the total weight (typically a majority, or two thirds) is owned by the honest parties.

Current blockchain protocols are proven secure assuming in addition a *multicast* network, which allows each party to distribute a value among the parties within some delivery time Δ. While provably-secure blockchain protocols have been proposed assuming an ideal multicast network (see e.g. [GKL15, PS17a, DGKR18, PS18, CM19, DPS19, AMN+20, DMM+20]), very little attention has been devoted to the construction of provably-secure multicast networks themselves.

In practice, the multicast network is typically implemented via a message-diffusion mechanism, where in order for a party P to distribute a message, P sends the message towards a subset of its neighbors, who then forward the message to their neighbors and so on. The idea is that if the honest parties remain connected, the message will reach all the honest parties after sufficiently many iterations. Intuitively, this works if the induced graph from the honest parties is connected. Indeed, there have been works that study how to randomly select the neighbors so that the induced graph remains connected (see e.g. [KMG03, RT19, MNT22]).

Unfortunately, to the best of our knowledge, currently analyzed diffusion mechanisms consider only the setting where all parties have the same weight, and can only be proven secure when a certain constant fraction of the honest parties (not a fraction of the total weight owned by honest parties) is honest.

This has two main drawbacks. First, when such a message diffusion mechanism is used to build a blockchain, the overall protocol relies on *both* the constant-honest-weight assumption **and** the constant-honest-fraction-of-parties assumption.

Second, the message-diffusion protocol does not take advantage of the actual distribution of weights in terms of efficiency. In fact, the incurred communication complexity of current multicast protocols is inversely proportional to the maximum number of tolerated corruptions: the higher the corruption threshold is, the lower the efficiency (intuitively, to guarantee security, each party needs to send to more neighbors). In particular, this means that in many of the current weight distributions where there are very few people owning a large fraction of the total weight, but thousands of parties owning a tiny little fraction of the weight, the incurred concrete communication complexity to achieve security significantly blows up (see an example in Figure 1, where even for large sizes of neighborhood sizes, the protocol fails).

The need for an efficient multicast network assuming solely the constant-honest-weight assumption is therefore apparent.

1.1 Our Contributions

In this work, we investigate provably-secure protocols that implement a multicast network for the weighted setting, relying solely on the constant-honest-weight assumption. We aim to achieve protocols with as low communication complexity as possible, and ideally, our protocols should incur no overhead with respect to the case where all parties own the same weight.

Is there a provably-secure multicast protocol in the weighted setting, assuming only a constant fraction of honest weight? And if so, is there any efficiency overhead with respect to the equal-weight setting?

Provably secure protocol. We show the first multicast protocol WFF (weighted fan-out flooding) that relies solely on the constant-honest-weight assumption. By naturally assigning the weights to corresponding stake quantities, the achieved guarantees match those required in previous proof-of-stake blockchain protocols (see e.g. [DGKR18, CM19, DMM+20]), and therefore our protocol can be used to build a blockchain protocol from point-to-point channels without the need for any additional assumption apart from those needed in the blockchain protocol itself. Moreover, our protocol incurs no overhead in the communication complexity compared to state of the art message-diffusion mechanisms designed for the equal-weights setting.

Theorem 1 (Informal)**.** *Let κ be a security parameter, n be the number of parties, and $\gamma \in [0,1]$ be the fraction of total weight belonging to honest parties. Further, let $\delta_{Channel}$ be an upper bound on the channel delays. Then, WFF is a secure flooding protocol with time complexity $\Delta := \left(7 \cdot \log\left(\frac{6n}{\log(n)+\kappa}\right) + 2\right) \cdot \delta_{Channel}$ and communication complexity $\frac{2n(\log(n)+\kappa)}{\gamma}$.*

Note that, in particular, the theorem shows that the expected communication complexity does not depend on the weight distribution.

Asymptotic optimality and practicality. Our protocol has the property that 1) parties accumulating large amounts of weight need to send to more parties, and 2) the number of parties that each party sends to increases logarithmically in the total number of parties. We prove that both properties are inherent for secure flooding protocols, meaning that Theorem 1 is asymptotically optimal. Concretely, for the first point, if a small set S (say, of constant size) accumulates more than a γ-fraction of the weight, then this set necessarily needs to send at least to a linear number $\Theta(n)$ of parties.

This means it is undesirable to have parties with very small weight and also to have parties with a huge weight. A simple way to mitigate this in practice is to exclude parties with less than $\alpha_{\min}$ weight and cap the maximal weight to $\alpha_{\max}$. This means if we use the flooding for a proof-of-stake blockchain, that parties with a huge amount of stake need to split their stake over several nodes such that none has more than $\alpha_{\max}$ weight. Parties with very little stake can still obtain data from other nodes by requesting data from them periodically. See Section 4 for more details.

Simulations. We use simulations to evaluate the practicality of our provably secure protocol. Our simulations show that the protocol is practical: not only are messages diffused quickly, but the communication complexity is also low, and this is regardless of both the weight distribution and of the adversary strategy. In fact, this goes in line with our theoretical results. More concretely, our simulations show our protocol guarantees the delivery of messages with high success probability even when weights are unevenly distributed. In fact, we also show that for equivalent neighborhood sizes, prior protocols—oblivious of the parties' weights—fail, whereas our provably secure protocol succeeds (See Figure 1). This in particular means that our protocol achieves the necessary security guarantees at a much lower communication cost than current (weight-oblivious) protocols, meaning that our protocol gives security for free.

1.2 Technical Overview

Below we elaborate on the technical contributions. Our protocol follows the structure of previous flooding protocols, where each party p samples a set of neighbours from the party set $\mathcal{P}$, according to some distribution $N \xleftarrow{\$} \mathcal{N}_p$. Then, upon receiving a message (Send, m) for the first time, it forwards the message to all parties in N.

A fundamental question. Before we dive into how to choose the neighborhood distribution, let us first show why some natural approaches fail to be secure in the weighted setting.

It is clear that to achieve efficient results, one must make use of the overall weight distribution to decide whether a party p_i forwards the message to party p_j. What is perhaps less clear, is what the required *amount* of dependency is. We start by arguing intuitively that the neighborhood selection must depend (at least) on both the weights of p_i and p_j.

Dependency on p_i's weight. Consider a weight distribution where p_i's weight is overwhelming, and there are many parties with very little weight (including p_j). In this case, the adversary has corruption budget to corrupt all parties except for p_i. Therefore, in order to guarantee that an honest p_j receives the message, p_i must send to all parties. That is, the neighborhood selection distribution must depend on p_i's weight.

Dependency on p_j's weight. In a similar fashion, if p_j's weight is overwhelming, an honest p_i must send his message to p_j, because there may be another honest p_k with very little weight, and therefore the protocol must ensure that p_j obtains the message so that he can relay it to all parties with small weights. That is, the neighborhood selection distribution must depend on p_j's weight.

A simple inefficient solution. From the above observations, we see that the neighborhood distribution must depend on both the weights W_i from p_i and W_j from p_j.

A simple idea is to let each party to internally emulate as many nodes as his weight, and then run a traditional flooding protocol among $W = \sum_i W_i$ nodes, where two nodes are connected with some probability ρ. By properties of Erdős–Rényi graphs [MNT22], this leads to a secure flooding protocol. Note that implicitly, the probability that a node from p_i is connected to a node from p_j depends on both weights W_i and W_j.

However, the resulting protocol is highly inefficient, since it has a communication complexity that depends on the total sum of the weights W, rather than the number of parties. Note that in current proof-of-stake systems, the total stake is in the order of billions, so any dependency on the total weight is highly undesirable.

Additionally, from traditional graph-theoretic results, it is not trivial to see whether one can remove edges from the resulting Erdős–Rényi graph, while at the same time maintaining the security of the protocol.

A first theoretical protocol. Our first technical theoretical contribution is a new simple way to choose the neighbours in the flooding protocol. More precisely, we generalize the approach above and show that it is actually enough to emulate a number of nodes that is proportional to the total number of parties (rather than the total weight).

For that, we introduce the notion of an emulation-function $\mathtt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$. According to the emulation function, we let each party p to internally emulate $\mathtt{E}(p) \geq 1$ different nodes, in a graph consisting of $n_{\mathtt{E}} = \sum_p \mathtt{E}(p)$ nodes. As explained above, the basic idea is to create an Erdős–Rényi graph on the emulated graph with $n_{\mathtt{E}}$ nodes and edge-probability ρ. Then, we say that a party p_i forwards the message to p_j if *any* of the emulated nodes from p_i is connected to *any* of the emulated nodes from p_j. This means that the probability that p_i forwards the message to p_j is $\rho_{(i,j)} = 1 - (1-\rho)^{\mathtt{E}(p_i)\cdot\mathtt{E}(p_j)}$.

Assuming that the honest fraction of total weight is γ, and that each party p emulates $\mathtt{E}(p) = \lceil \alpha_p \cdot n \rceil$ nodes, where α_p is p's fraction of the total weight he owns, we obtain a flooding protocol with communication complexity $O((\log(n)+\kappa)\cdot n \cdot \gamma^{-1})$ and time complexity $O(\log(n)\cdot\delta_{\text{Channel}})$.

A practical protocol. Although the method described above is intuitive and gives us asymptotically good complexities, it is very far from being practical. In particular, the protocol requires locally flipping n coins for each message sent and/or forwarded.

Similar to current protocols deployed in practice, we would like to have a protocol that instead chooses a *fixed* set of neighbors (possibly dependent on the weight distribution, but nothing else), and provide provable security for it.

We propose a protocol where each party p chooses to send to $K = k \cdot \mathtt{E}(p) = k \cdot \lceil \alpha_p \cdot n \rceil$ distinct parties (for a parameter k), according to a weighted sampling without replacement [BHPS16]. More precisely, p chooses the parties in an ordered manner, and the probability to choose a certain ordered tuple of parties $(q_1, \ldots, q_K)$ (among the set of parties $\mathcal{P} \setminus \{p\}$) is defined as follows:

$$\Pr\left((q_1, \ldots, q_K)\right) = \prod_{i=1}^{K} \frac{\mathtt{E}(q_i)}{n_{\mathtt{E}} - \mathtt{E}(p) - \mathtt{E}(q_1) - \cdots - \mathtt{E}(q_{i-1})}.$$ [1]

Note that even though this protocol is so simple that it can be described in just a single line, it is by no means trivial. In fact, it is crucial for the correctness of the protocol that the emulation function is used to determine both the number of neighbours *and* the distribution of these neighbours.

To see that it is crucial to use the emulation function to decide how many neighbours each party should choose, consider a small change to the protocol, namely send to $K = k \cdot \alpha_p \cdot n$ parties (instead of $K = k \cdot \mathtt{E}(p)$). Now, consider sender p with a small fraction of the total weight α_p, and let us estimate the parameter k to ensure that this p sends to at least one honest party. As any party potentially could be corrupt it must be that p sends to more than just one neighbour. Hence, it must be that $k > \frac{1}{\alpha_p \cdot n}$, just to ensure this very minimal requirement. A rough bound on the communication complexity of such protocol would be $\sum_{p'} k \cdot \alpha_{p'} \cdot n > \frac{1}{\alpha_p}$, which is impractical if α_p is small.

To see that it is crucial to weigh the selection of neighbours with the emulation function we consider another small change to the protocol, namely to select parties weighted according to their weight instead of the emulation function. Now, consider a weight distribution where there is just one party p with a very small fraction of the total weight and all others having roughly the same weight. Note, that for any party choosing less than n neighbours the probability that p is chosen as a neighbour becomes arbitrarily small for a decreasing α_p. Hence, to ensure that someone sends to p this would induce a quadratic communication complexity which is impractical.

Proving security of such a protocol in the weighted setting directly is non-trivial for two reasons: First, the choices of whether to send to a neighbour or not are not independent. Second, the fact that the choices are according to an arbitrary weight distribution makes the analysis considerably harder than traditional graph-theoretic results that consider the non-weighted setting. Instead of providing a direct graph-theoretic analysis, we give a security proof via a sequence of intermediate protocols, essentially relating the success probability of the first protocol above based on Erdős–Rényi graphs, to the practical protocol. This leads to Theorem 1.

Evaluation. We evaluate the practicality of our proposed protocol by running various simulations to measure its security[2], its scalability, and to estimate what protocol parameters

[1] The probability to choose neighborhood set $N = \{q_1, \ldots, q_K\}$ is the sum over the probabilities of all permuted tuples.

[2] Here, security means not only whether all parties get the message, but also how long it takes for all parties to have received the message.

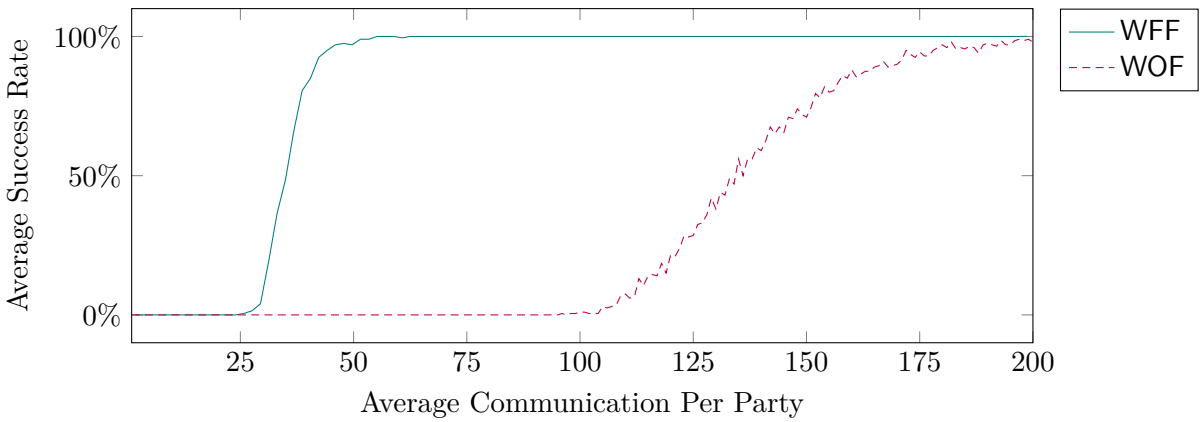


Figure 1: Plot extracted from Figure 2. In the plot, WFF denotes our proposed protocol, whereas WOF denotes a weight oblivious flooding protocol where each party forwards each message to a fixed number of parties sampled uniformly. For the plot, we consider 1024 parties, an exponential weight distribution (where the heaviest party's weight is 1.000.000 times the lightest party's weight), a corruption threshold of 50%, and a greedy corruption strategy where lighter parties are corrupted first. For each configuration, we make 200 runs.

achieve security in practice. Our simulations show that our protocol can achieve security for realistic weight distributions whereas the weight-oblivious flooding protocols currently used by blockchains do not (as illustrated by the results plotted in Figure 1), and that it is also extremely scalable. Our evaluations consider various parameters such as the number of parties, the weight distributions, who the sender is, corruption strategies, corruption thresholds, and more.

1.3 Current State of the Art and Related Work

Flooding networks in a Byzantine setting. [KMG03] was the first to relate probabilistic gossiping to the connectivity of the induced graph. They considered $(1-\gamma)\cdot n$ out of n parties failing and showed that each party needs to forward a message with probability $\rho > \frac{\log(n)+\kappa}{\gamma \cdot n}$ to any other party in order to ensure that messages are delivered to all non-failing parties with a probability overwhelming in κ.

[MNT22] observed that against an adversary capable of adaptively corrupting up to t parties, any flooding network where each party sends to less than t neighbours is inherently insecure (an adversary can simply corrupt all neighbours of a sender). To mitigate this problem and achieve a protocol secure against a Byzantine adaptive adversary, [MNT22] introduced the notion of a delayed adversary for which there is a delay from the time the adversary decides to corrupt a party until the party is effectively controlled by the adversary. In this setting, they showed that against an adversary delayed for the time it takes to send a message plus the time it takes to resend a message, it is sufficient to on average send to $\Omega((\log(n)+\kappa)\cdot\gamma^{-1})$ neighbours to achieve a gossip protocol that with an overwhelming probability in κ has $O(\log(n))$ round complexity for n parties with at most $(1-\gamma)\cdot n$ of the parties being corrupted. In this work, we match the theoretical performance of their flooding protocol with a practical protocol that relies on the assumption that only a γ fraction of the weight remains honest, which is more relevant in the blockchain setting. Furthermore, we believe the techniques from [MNT22] can be applied to also show that our protocol is secure against such delayed adaptive adversaries.

Kadcast [RT19] is a recent flooding protocol specifically designed for blockchains. Interestingly, they claim that structured networks are inherently more efficient than unstructured

networks and propose a structured protocol with $O(\log n)$ neighbours and $O(\log n)$ steps to propagate a message, which is similar to what we achieve using an unstructured network. It is unclear how their protocol performs under Byzantine failures. Further, we note that structured networks are inherently vulnerable to attacks by adaptive adversaries.

A different line of work [MMR99, MPS01, MS03] considers how to propagate updates in a database using gossip where at most t of the processors may be corrupted. The setting is however different from ours as they assume that at least t honest parties get the update as input initially, and only updates input to some honest processor can be accepted by the other processors.

Probabilistic communication have also been used to improve the communication complexity for both multi-party-computation (MPC) [CCG+15] and Byzantine broadcast [TLP20]. In [CCG+15], communication between honest parties is assumed to be hidden from the adversary. This is exploited by constructing a random communication network with an average polylogarithmic degree based on Erdős–Rényi graphs. They thereby achieve a MPC protocol with low communication locality that is secure against a fully adaptive adversary. [TLP20] combines the classic broadcast protocol by Dolev and Strong [DS83] with gossiping based upon Erdős–Rényi-graphs to obtain the first broadcast algorithm with a sub-cubic communication complexity for a dishonest majority. Using similar techniques and assuming a trusted setup they also achieve an asymptotically optimal communication complexity for parallel broadcast.

Attacks on the network layers of blockchains. Attacks on network layers of blockchains are not only a theoretical concern. In fact, several works [HKZG15, AZV17, MHG18, TCM+20] have shown that it has been possible to launch eclipsing attacks against nodes in the Bitcoin network and the Ethereum network.

Bitcoin's peer-to-peer network works by letting each node in the network maintain 8 outgoing connections and up to 117 incoming connections. This is clearly insecure when considering a resource-constrained adversary instead of a traditional adversary (as the probability of only connecting to adversarial nodes can be arbitrarily high). Additional to this inherent insecurity, [HKZG15] showed how to eclipse a node that is already a part of an existing honest network by exploiting a bias in the way a peer selects its outgoing connections. They launched such an attack with only 4600 bots and achieved 85% success probability to actually eclipse a targeted node.

By default, a node in the Ethereum peer-to-peer network selects 13 outgoing connections contrary to the 8 that is the default in Bitcoin. Hence, one might be led to believe that it is more difficult to eclipse an Ethereum node than a Bitcoin node. However, in a Ethereum neighbours are selected using a distance measure that is based on nodes' public keys. Exploiting that in a prior version of the Ethereum client a single computer was allowed to run several nodes, [MHG18] showed that just a single computer can be used to mount an attack by creating multiple carefully selected public keys.

[AZV17] showed that BGP-Hijacking can also be used to eclipse Bitcoin nodes. However, we note that such attack is immediately observable as an adversary will need to announce a false BGP prefix publicly. In [TCM+20], it was shown that a stealthier version of such an attack in can also be launched against a Bitcoin node by additionally influencing how a bitcoin node selects its outgoing connections. We note that such attacks are attacks on the infrastructure of the internet, and therefore fall outside the scope of our model.

We note that the attacks presented in [HKZG15, MHG18, TCM+20] all rely on exploiting the heuristics used to select outgoing connections for nodes in the peer-to-peer network. Hence, such attacks would not have been possible if, instead of heuristics a provably secure protocol (such as the one presented in this work) had been deployed.

Detecting eclipse attacks. As a way of mitigating attacks on the network layer a line of work considers the possibility of detecting eclipse attacks [XGS+20, ZTA21, ARV+21]. [XGS+20] provide a method for using supervised learning to detect eclipsing attacks based on the metadata in packages. We note that this method is only as good as its data set for training, and hence cannot be used to detect attacks in general. A different approach is to try to detect eclipse attacks based on the absence of new blocks [ZTA21, ARV+21]. However, this method has the drawback that it becomes arbitrarily slow as the fraction of resources controlled by an adversary approaches 50%, and even for small values, it takes upwards of 3 hours to detect. Finally, it has been considered to detect eclipse attacks using an additional overlay gossip protocol [ARV+21]. However, contrary to this work this is not *proven* to work but rather demonstrated to work empirically.

Consequences of eclipse attacks. If a party is eclipsed it is immediate that security proofs that rely on guaranteed message delivery no longer apply. Several works have shown that eclipse attacks do not only invalidate the security proofs but actually invalidate the actual security of blockchain protocols [HKZG15, NKMS16, ZL19]. Eclipsing can be used to invalidate the total order that blockchain provides and thereby allow double-spend attacks [HKZG15], amplify the rewards from selfish mining [NKMS16], and dramatically speed up "stake-bleeding"-attacks [ZL19].

1.4 Outline of the Paper

In Section 2 we introduce the model for which our results hold as well as introduce notation and basic graph definitions that are used in the remainder of the paper. In Section 3 we present our practical flooding protocol and prove it secure based on the honest-weight assumption. In Section 4, we present theoretical lower bounds showing that our protocol is asymptotically optimal, as well as discuss practical implications of these bounds. In Section 5, we present two solutions for obtaining delivery to parties with zero weight. Finally, in Section 6 we show our practical simulations.

2 Notation and Model

2.1 Notation

We will use κ to denote the security parameter of our protocols. We will write $A \xleftarrow{\$} \mathcal{D}$ to sample the value A from the distribution $\mathcal{D}$ and use the infix notation $\sim$ to denote that two random variables are distributed identically. We will let $\mathcal{B}(n, \rho)$ denote the binomial distribution with parameters n and ρ, and $\mathcal{U}(A)$ denote the uniform distribution on a set A. We denote by $\log x$ the natural logarithm of x.

In our proofs we will write RHS and LHS to refer to respectively the right hand side and left hand side of (in)equalities.

2.2 Parties, Weight, Adversary and Communication Network

We let $\mathcal{P}$ denote the static set of parties for which our protocols will work. For convenience we let $n \coloneqq |\mathcal{P}|$ and let $\mathcal{H} \subseteq \mathcal{P}$ be the set of parties that are honest.

We assume that a public weight is assigned to each party. We let W_p denote the weight assigned to party p, and let $\alpha_p \coloneqq \frac{W_p}{\sum_{p \in \mathcal{P}} W_p}$ i.e., the fraction of the total weight assigned to party p.

We allow an adversary to corrupt any subset of the parties such that the remaining set of honest parties together constitutes more than a $\gamma \in (0, 1]$ fraction of the total weight. Formally, we assume that

$$\sum_{p \in \mathcal{H}} \alpha_p \geq \gamma, \tag{1}$$

and that all parties have a non-zero positive weight i.e. $\forall p \in \mathcal{P}, W_p > 0$.[3] We will refer to this assumption as the honest weight assumption. For simplicity, we consider a static adversary, although our results also hold against a so-called delayed-adaptive adversary [MNT22], where the corruptions can be adaptively chosen but only happen after a certain amount of time.

Parties $\mathcal{P}$ have access to a complete network of point-to-point authenticated channels that ensures delivery within a bounded delay. Concretely, we assume that all channels ensures delivery within δ_{Channel} time.

Realising public weights from resource assumptions. Proof-of-stake blockchains [DGKR18, DPS19, CM19] rely on a constant fraction of the stake being honest (more than $1/2$ for [DGKR18, DPS19] and more than $2/3$ for [CM19]). Furthermore, a blockchain itself provides a ledger accessible by all parties describing how much stake each party owns. Hence, it is immediate how to assign weights to parties by simply accessing the ledger in order to instantiate the weights for our protocols. For simplicity, we do not consider the dynamic stake setting where stake can change throughout the execution. This is however not a real limitation of our protocol. Take for example [DGKR18]. In order to prove their protocol secure for a dynamic stake, they divide time into epochs where the stake used for producing blocks remains unchanged and additionally make assumptions on the speed that stake can between epochs. In their proofs, they note that all parties agree on the stake distribution in a previous epoch. We note that the time it takes for our protocols to propagate a message is very small compared to such epochs, and furthermore our proofs only rely on the weight being static for the propagation of a single message.

To achieve a weight distribution for blockchain protocols that rely on a constant fraction of the computational resources being honest [GKL15, PS17a, PS17b, PS18] one can make use of the techniques for committee selection for such setting [PS17b, PS18]. The idea behind this is that for long fragments of a chain with high chain-quality, the distribution of block creators is similar to the distribution of computational resources among parties. Hence, this distribution translates directly to a weight distribution publicly available to all parties. For techniques to achieve a high chain-quality see [PS17a].

2.3 Graphs

We use standard notation for a undirected and directed (di)graphs.

Definition 1 ((di)Graph)**.** A (di)graph consists of a set of vertices $\mathcal{V}$ and a set of edges $E \subseteq \mathcal{V} \times \mathcal{V}$. For an undirected graph we interpret the edge $\{v, z\}$ as being an undirected edge between v and z. For a digraph we interpret the edge (v, z) as being a directed edge from v to z. For a graph G with nodes $\mathcal{V}$ and edges E we write $G = (\mathcal{V}, E)$.

Definition 2 (Directed graph)**.** A directed graph (digraph) consists of a set of vertices $\mathcal{V}$ and a set of edges $E \subseteq \mathcal{V} \times \mathcal{V}$. For two vertices $v, z \in \mathcal{V}$ we interpret the edge (v, z) as being a directed edge from v to z. For a digraph G with nodes $\mathcal{V}$ and edges E we write $G = (\mathcal{V}, E)$.

[3] For a discussion of the necessity of the zero-weight requirement see Section 4 and for methods to anyway achieve delivery to such zero-weight parties see Section 5.

Additionally, we define notation for distance and diameter properties which we overload to work for both directed and undirected graphs.

Definition 3 (Distance)**.** For a (di)graph $G = (\mathcal{V}, E)$ we let the distance function be denoted by $\texttt{dist} : \mathcal{V} \times \mathcal{V} \to \mathbb{N}$. The distance between two nodes $v, z \in \mathcal{V}$, written $\texttt{dist}(v, z)$, is defined as the length of shortest path between any v and z. If no such path exists we define $\texttt{dist}(v, z) \triangleq \infty$.

Definition 4 (Low Distance)**.** For $\lambda \in \mathbb{N}$ and a (di)graph $G = (\mathcal{V}, E)$ we let $\phi_{\text{Dist}}(G, v, \lambda)$ be the property that the distance from $v \in \mathcal{V}$ is less than λ. I.e. we define

$$\phi_{\text{Dist}}(G, v, \lambda) \triangleq \bigwedge_{z \in \mathcal{V}} \texttt{dist}(v, z) \leq \lambda.$$

Definition 5 (Low Diameter)**.** For $\lambda \in \mathbb{N}$ and a (di)graph $G = (\mathcal{V}, E)$ we let $\phi_{\text{Diam}}(G, \lambda)$ be the property that the diameter of G is less than λ. I.e. we define

$$\phi_{\text{Diam}}(G, \lambda) \triangleq \bigwedge_{v \in \mathcal{V}} \phi_{\text{Dist}}(G, v, \lambda).$$

We finally define the distributions of both undirected and directed Erdős–Rényi graphs.

Definition 6 (Erdős–Rényi (di)graphs)**.** An Erdős–Rényi (di)graph is an (di)graph $G = (\mathcal{V}, E)$ where all possible edges are present with an independent probability ρ. That is for any $v, z \in \mathcal{V}$, we have $\Pr[\{v, z\} \in E] = \rho$ for Erdős–Rényi graphs and $\Pr[(v, z) \in E] = \rho$ for digraphs. To sample such a graph G with $|\mathcal{V}| = \eta$, we write $G \xleftarrow{\$} \mathcal{G}_{\text{ER}}(\eta, \rho)$ and for the directed case $G \xleftarrow{\$} \mathcal{G}_{\overrightarrow{\text{ER}}}(\eta, \rho)$.

3 Weighted Flooding

In this section we present a practical and provably secure flooding protocol that only relies on the honest weight assumption. Before doing so we first present our definition of a flooding protocol in Section 3.1. Then, in Section 3.2 we present a generic skeleton for flooding protocols that is parameterized by the way parties selects their neighbours and prove that it is sufficient to consider the way neighbours are selected in order to get a secure protocol. We use this skeleton to define our theoretical flooding protocol that is secure based upon each party emulating a number of nodes proportional to their weight in an Erdős–Rényi graph (Section 3.3). Finally, in Section 3.4 we present our practical protocol and prove that it is secure.

3.1 Properties of Flooding Protocols

Below we give our property based definition of a flooding protocol.[4]

Definition 7. Let Π be a protocol executed by parties $\mathcal{P}$, where each party $p \in \mathcal{P}$ can input a message at any time, and as a consequence all parties get a message as output. We say that Π is a Δ-flooding protocol if the two properties hold with a probability overwhelming in the security parameter κ for each message m:

1) If m is input by an *honest* party for the first time at time τ, then by time $\tau + \Delta$ it is ensured that all other honest parties output m.

[4]Note that for protocols with no secrecy (each event is leaked to the adversary), and for functionalities that give the adversary full control while respecting these properties a simulation-based security notion is directly implied by the property-based definition. For flooding networks, this technique is used in the proofs in [MNT22].

2) If m is output by an *honest* party at time τ, then by time $\tau + \Delta$ it is ensured that all honest parties output m.

Note that this definition subsumes the assumptions that most blockchain protocols rely on [GKL15, PS17a, DGKR18, PS18, CM19, DMM+20]. To the best of our knowledge only [DMM+20] relies on both Properties 1) and 2), whereas the other works only rely solely on Property 1). However, as Property 2) essentially comes for free for the type of protocols we consider (each party will forward everything they receive and thereby act as if they themselves send the message) we have chosen to include it in our definition. Furthermore, because of this structure of our protocols, it is sufficient to bound the probability of Property 1) in order to show that our protocols are in fact flooding protocols according to the definition. For our proofs and lemma statements, it is, therefore, useful to define notation for the predicate that a message input to an honest party for the first time is delivered respecting the delivery bound for a flooding protocol, which is what we encapsulate in the predicate below.

Definition 8 (Timely delivery)**.** For a message m that is input for the first time at an honest party at time τ we say that m is Δ-*timely-delivered* if all honest parties have output m no later than time $\tau + \Delta$. We let $\texttt{Timely}_m(\Delta)$ denote the induced predicate.

3.2 A Skeleton For Flooding Protocols

We now present a skeleton for our flooding algorithm. The structure of the protocol is very similar to the protocols proposed in [MNT22], but contrary to their protocols our protocol takes an additional parameter $\mathcal{N}$, which is an algorithm that allows each party to sample a set of neighbours. We refer to this parameter as the *neighbourhood selection algorithm*.

The protocol accepts two commands: One for sending and one for checking which messages have been received. Once a send command is issued to a party, the party will forward the message to a set of neighbours that are determined using the neighbourhood selection algorithm. Furthermore, once a message is received on a point-to-point channel the receiver checks if the message has already been relayed and if not it forwards the message to a set of neighbours that is again selected using the neighbourhood selection algorithm.

Protocol $\pi_{\text{Flood}}(\mathcal{N})$

We use $\mathcal{N}_p$ to denote the neighbourhood distribution of party p. Each party $p_i \in \mathcal{P}$ keeps track of a set of relayed messages $\texttt{Relayed}_i$ which will also be used to keep track of which messages party p_i has received.

Initialize: Initially, each party p_i sets $\texttt{Relayed}_i \coloneqq \varnothing$.

Send: When p_i receives (Send, m), they sample a set of neighbours $N \xleftarrow{\$} \mathcal{N}_p$ and forwards the message to all parties in N. Finally, they set $\texttt{Relayed}_i \coloneqq \texttt{Relayed}_i \cup \{m\}$.

Get Messages: When p_i receives (GetMessages) they return $\texttt{Relayed}_i$.

When party p_i receives message m on a point-to-point channel where $m \notin \texttt{Relayed}_i$, p_i continues as if they had received (Send, m). Otherwise it ignores m.

Our protocols will take the structure of π_{Flood}, but the neighbourhood selection algorithms will vary. Hence, we would like to be able to relate the security of the overall flooding protocol to just the neighbourhood selection algorithm used. To do so we first define a random process for

creating a graph where each honest party is a node, given a family of neighbourhood selection algorithms $\mathcal{N}$, a starting party p, and a distance λ. The intuition is that this process mimics the worst-case behavior of the adversary during a sending process starting from party p. However, separating this into a process without adversarial influence allows us to relate probabilistic experiments without taking into account the choices of an adversary which could have a strategy that depends on parts of the outcome of the experiments.

Definition 9. Let $\mathcal{N}$ be a family of neighbourhood selection algorithms, let $p \in \mathcal{H}$, and let $\lambda \in \mathbb{N}$ be a distance. We let the *honest sending process*, $\mathsf{HSP}(p, \mathcal{N}, \lambda)$, be a random process that returns a directed graph $G = (\mathcal{V}, E)$ defined by the following random procedure:

1. Initially, $E := \varnothing$. Furthermore, we keep track of set $\texttt{Flipped} := \varnothing$ that consists of nodes that have already had their outgoing edges decided, and a first-in-first-out queue $\texttt{ToBeFlipped} := \{(p, 0)\}$ of nodes and their distance from p that are to have their edges decided.
2. The process proceeds with the following until $\texttt{ToBeFlipped} == \varnothing$.
   (a) Take out the first element of $\texttt{ToBeFlipped}$ and let it be denoted by (p', i).
   (b) Let $N \xleftarrow{\$} \mathcal{N}_{p'}$ and set $N := N \cap \mathcal{H}$.
   (c) Now, update the set of edges $E := E \cup \{(p', p'') \mid p'' \in N\}$ and let $\texttt{Flipped} := \texttt{Flipped} \cup \{p'\}$.
   (d) Furthermore, if $i+1 < \lambda$ then for all $p'' \in N \setminus \texttt{Flipped}$ add $(p'', i+1)$ to $\texttt{ToBeFlipped}$.
3. Finally, return $G = (\mathcal{H}, E)$.

Next, we are interested in bounding the probability that a message is delivered within the time guaranteed by the flooding algorithm in terms of the probability that there is a low distance to all parties from the sender. We show that the probability that π_{Flood} ensures timely delivery for a message is lower-bounded by the probability that the honest sending process results in a graph where the sender can reach all other honest nodes within a certain number of steps.

Lemma 1. *Let $\mathcal{N}$ be a family of neighbourhood selection algorithms, let $p \in \mathcal{H}$, and let $\lambda \in \mathbb{N}$ be a distance. Further, let m be a message that is input to p for the first time during the execution of $\pi_{Flood}(\mathcal{N})$ and let $G \xleftarrow{\$} \mathsf{HSP}(p, \mathcal{N}, \lambda)$. Then,*

$$\Pr[\phi_{Dist}(G, p, \lambda)] \leq \Pr[\texttt{Timely}_m(\lambda \cdot \delta_{Channel})]. \tag{2}$$

Proof. To see this we consider a another graph $G' = (\mathcal{V}', E')$ where each honest party is a node (i.e., $\mathcal{V}' = \mathcal{H}$) and there is an edge $(p_i, p_j)E'$ from party p_i to p_j if and only if p_j received the message m from p_i before time $\tau + \lambda \cdot \delta_{\text{Channel}}$. In this graph, it is clear that the delivery guarantee is satisfied for any node with an incoming edge. In particular if $\phi_{\text{Dist}}(G', p, \lambda)$ then $\texttt{Timely}_m(\lambda \cdot \delta_{\text{Channel}})$. It is hence sufficient to argue that $\phi_{\text{Dist}}(G', p, \lambda)$ stochastically dominates $\phi_{\text{Dist}}(G, p, \lambda)$. We show this via a straightforward coupling between the two graphs via a graph $\widetilde{G} = (\mathcal{H}, \widetilde{E})$, namely by first constructing G' and then construct $\widetilde{G}$ by duplicating the edges from E' for all parties within distance $\lambda - 1$ of p to also appear in $\widetilde{E}$. It is clear that the $\widetilde{E} \subseteq E'$ and hence that $\phi_{\text{Dist}}(\widetilde{G}, p, \lambda) \implies \phi_{\text{Dist}}(G', p, \lambda)$. Therefore, what is left is only to argue that $G \sim \widetilde{G}$. Observe that any node that receives the message at the latest at $\tau + (\lambda - 1) \cdot \delta_{\text{Channel}}$ is ensured to have edges added corresponding to $\mathcal{N}$ in G' and hence also to $\widetilde{G}$. Furthermore, for any $d \in \mathbb{N}$ at time $\tau + d \cdot \delta_{\text{Channel}}$ any node at distance d from the sender will get the message delivered and thereby select their neighbours. Therefore all nodes at distance $\lambda - 1$ will have all their edges added to the graph $\widetilde{G}$ and hence $G \sim \widetilde{G}$. □

Lemma 1 ensures that it is sufficient to consider neighbourhood selection algorithms and prove that graphs constructed via. the honest sending process has a low distance from the sender to all other parties.

3.3 Emulating Nodes in Erdős–Rényi Graphs

Our central idea for achieving a flooding network that relies on the honest weight assumption is to let each party emulate a number of nodes proportional to their weight in a hypothetical Erdős–Rényi graph. We will refer to this hypothetical graph as the *emulated graph*. Now, our idea is that if there is an edge between an emulated node v and another emulated node z corresponds to that the party emulating node v should forward the message to the party emulating z. Our goal is now to ensure each honest party emulates at least one node and that the emulated graph has a low diameter, as this will result in that all parties will receive the message quickly.

Concretely, we introduce a function $\mathtt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$ which for each party determines how many nodes this party should act as in the emulated graph. We refer to this function as the *emulation function*. [5] For such emulation function we define notation for the number of emulated nodes $n_{\mathtt{E}}$ and the number of honest nodes that are emulated $h_{\mathtt{E}}$:

$$n_{\mathtt{E}} \triangleq \sum_{p \in \mathcal{P}} \mathtt{E}(p) \quad \text{and} \quad h_{\mathtt{E}} \triangleq \sum_{p \in \mathcal{H}} \mathtt{E}(p).$$

Before looking at how to choose an emulation function, let us present how the idea leads to a very simple algorithm for selecting neighbors by letting the emulated graph take the form of an Erdős–Rényi graph. We let ρ denote the probability that there should be an edge between any two nodes in the emulated graph. Now, the probability that party p_i should forward a message to party p_j is:

$$\begin{aligned}
&\Pr[p_i \text{ should forward a message to } p_j] \\
&:= \Pr[\text{there is an edge from any of } p_i\text{'s emulated nodes to any of } p_j\text{'s emulated nodes}] \\
&= 1 - \Pr[\text{there are no edges between any of } p_i \text{ and } p_j\text{'s emulated nodes}] \\
&= 1 - (1 - \Pr[\text{there is an edge between any two emulated nodes})^{\mathtt{E}(p_i)\cdot\mathtt{E}(p_j)} \\
&= 1 - (1-\rho)^{\mathtt{E}(p_i)\cdot\mathtt{E}(p_j)}.
\end{aligned} \tag{3}$$

This give rise to the following family of neighbour selection algorithms indexed by a party $p \in \mathcal{P}$ and parameterized by both an emulation function $\mathtt{E}$ and an edge probability ρ.

Algorithm ER-Emulation$_p(\mathtt{E}, \rho)$

Let $N := \varnothing$.
Let $P := \mathcal{P} \setminus p$.
while $P \neq \varnothing$ **do**
    Pick $r \in P$.
    Sample $c \xleftarrow{\$} \mathcal{U}([0,1])$.
    if $c \leq 1 - (1-\rho)^{\mathtt{E}(p)\cdot\mathtt{E}(r)}$ **then**
        Update $N := N \cup \{r\}$.
    end if
    Update $P := P \setminus \{r\}$.

[5] For a function to be an emulation function, we require that all parties should emulate at least 1 node, which is why the codomain of the function is defined to be $\mathbb{N} \setminus \{0\}$.

10: **end while**
11: **return** N.

Relating Erdős–Rényi graphs and the honest sending process. We now formalize the intuition that given that an emulated graph is "well connected" then the honest sending process results in a graph that is also "well connected". In particular, we show a relation between the probability that the distance in a directed Erdős–Rényi graph of size similar to the size of the honest emulated graph is large to the probability that the distance from the sender is large in the honest sending process.

Lemma 2. *Let $\rho \in [0,1]$, let $\lambda \in \mathbb{N}$, let $p \in \mathcal{H}$, and let $\mathtt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$ be an emulation function. Further, let $G_1 \xleftarrow{\$} \mathsf{HSP}(p, \mathsf{ER\text{-}Emulation}(\mathtt{E}, \rho), \lambda)$ and let $G_2 \xleftarrow{\$} \mathcal{G}^{\rightarrow}_{\mathrm{ER}}(h_{\mathtt{E}}, \rho)$. Then for any node $v \in \mathcal{V}$ we have,*

$$\Pr[\phi_{Dist}(G_2, v, \lambda)] \leq \Pr[\phi_{Dist}(G_1, p, \lambda)]. \tag{4}$$

Proof. Let $v \in \mathcal{V}$. We define a simple coupling between the two graphs, G_1 and $G_2 = (\mathcal{V}, E_2)$, via a new graph $\widetilde{G_1} = (\mathcal{H}, \widetilde{E_1})$.

Before defining the coupling itself we define some additional notation. We define a mapping $\mathtt{m} : \mathcal{H} \to 2^{\mathcal{V}}$ that maps each honest party to a set of nodes via. the emulation function. That is for each party $p' \in \mathcal{H}$ we define $\mathtt{m}(p')$ to be a set of parties $\left\{ z_{p'_1}, \ldots, z_{p'_{\mathtt{E}(p')}} \right\}$ s.t. for any two parties $p_i, p_j \in \mathcal{H}$ we have that their sets of emulated nodes are non-overlapping i.e., $\mathtt{m}(p_i) \cap \mathtt{m}(p_j) = \varnothing$. Further, we define it such that $v \in \mathtt{m}(p)$. Note that $\bigcup_{p' \in \mathcal{H}} \mathtt{m}(p') = \mathcal{V}$. Similarly, we will use $\mathtt{m}^{-1} : \mathcal{V} \to \mathcal{H}$ to find the party that "emulates" a particular node. We now define our coupling via. the following random process that mimics the honest sending process closely.

1. Initially, sample $G_2 = (\mathcal{V}, E_2) \xleftarrow{\$} \mathcal{G}^{\rightarrow}_{\mathrm{ER}}(h_{\mathtt{E}}, \rho)$ and let $\widetilde{E_1} \coloneqq \varnothing$. Furthermore, we keep track of set $\mathtt{Flipped} \coloneqq \varnothing$ that consists of nodes that have already had their outgoing edges decided, and a first-in-first-out queue $\mathtt{ToBeFlipped} \coloneqq \{(p, 0)\}$ of nodes and their distance from p that are to have their edges decided.

2. The process proceeds with the following until $\mathtt{ToBeFlipped} == \varnothing$.

   (a) Take out the first element of $\mathtt{ToBeFlipped}$ and let it be denoted by (p', i).

   (b) We now update find the neighbourhood of p' by looking at the edges from the emulated nodes of party p' i.e., we set $N \coloneqq \{(p', \mathtt{m}^{-1}(z)) \mid w \in \mathtt{m}(p') \wedge (w, z) \in E_2\} \setminus \{(p', p')\}$.

   (c) Now, update the set of edges $E \coloneqq E \cup \{(p', p'') \mid p'' \in N\}$ and let $\mathtt{Flipped} \coloneqq \mathtt{Flipped} \cup \{p'\}$.

   (d) Furthermore, if $i+1 < \lambda$ then for all $p'' \in N \setminus \mathtt{Flipped}$ add $(p'', i+1)$ to $\mathtt{ToBeFlipped}$.

3. Finally, return $\widetilde{G_1} = (\mathcal{H}, \widetilde{E_1})$.

That $G_1 \sim \widetilde{G_1}$ follows from Equation (3). It is thus left to show that $\phi_{\mathrm{Dist}}(G_2, v, \lambda) \implies \phi_{\mathrm{Dist}}(\widetilde{G_1}, p, \lambda)$. We prove this by proving the contrapositive statement, $\neg\phi_{\mathrm{Dist}}(\widetilde{G_1}, p, \lambda) \implies \neg\phi_{\mathrm{Dist}}(G_2, v, \lambda)$. Assuming the LHS of the implication we get that there exists some party $p' \in \mathcal{H}$ that is not within distance λ of p. Now let $z \in \mathtt{m}(p')$ be a node that is emulated by p' (we know that such exists by the definition of the emulation function) and let us show that z is not within distance of v. For the sake of contradiction assume that z in fact is within distance λ of

v. However, any path in G_2 of length at most λ from v induces a path in the $\widetilde{G_1}$ by applying $\mathtt{m}^{-1}$ to the path. When pruned for duplicate nodes this path in $\widetilde{G_1}$ is at most as long as the path in G_2, and hence we have a contradiction. □

Next, we show that the probability that a particular node can reach all other nodes within a certain distance in a directed Erdős–Rényi graph is lower-bounded by the probability that an undirected Erdős–Rényi graph has a high diameter.

Lemma 3. *Let $\rho \in [0,1]$, let $\lambda \in \mathbb{N}$ and let $\eta \in \mathbb{N}$. Further, let $G_1 = (\mathcal{V}_1, E_1) \xleftarrow{\$} \mathcal{G}^{\rightarrow}_{\mathrm{ER}}(\eta, \rho)$ and let $G_2 = (\mathcal{V}_2, E_2) \xleftarrow{\$} \mathcal{G}_{\mathrm{ER}}(\eta, \rho)$.*

Then for any node $v \in \mathcal{V}_1$ we have,

$$\Pr[\phi_{Diam}(G_2, \lambda)] \leq \Pr[\phi_{Dist}(G_1, v, \lambda)]. \tag{5}$$

Proof. We observe that $\mathcal{V}_1 = \mathcal{V}_2$ and will from now on just use $\mathcal{V}$. Let $v \in \mathcal{V}$. It is now sufficient to show that $\phi_{\mathrm{Dist}}(G_1, v, \lambda)$ stochastically dominates $\phi_{\mathrm{Diam}}(G_2, \lambda)$. We show this by defining a coupling of the two graphs $\widetilde{G_1} = (\mathcal{V}, \widetilde{E_1})$ and $\widetilde{G_2} = (\mathcal{V}, \widetilde{E_2})$ and show that if $\phi_{\mathrm{Diam}}(\widetilde{G_2}, \lambda)$ holds then also $\phi_{\mathrm{Dist}}(\widetilde{G_1}, v, \lambda)$ holds. The idea behind the coupling is to start an edge-selection process by selecting the edges of v, and now select nodes the next node to pick edges by taking the one that is "closest" to v. In more detail, we define the coupling via. the following random process.

1. Initially, $\widetilde{E_1} \coloneqq \widetilde{E_2} \coloneqq \varnothing$. Furthermore, we keep track of set $\mathtt{Flipped} \coloneqq \varnothing$ that consists of nodes that have already had their outgoing edges decided, and a first-in-first-out queue $\mathtt{ToBeFlipped} \coloneqq \{v\}$ of nodes that are to have their edges decided.
2. The process proceeds with the following loop until $\mathtt{Flipped} == \mathcal{V}$.
   (a) Take out the first of $\mathtt{ToBeFlipped}$ and call it z. If no such node exists let z be an arbitrary element in $\mathcal{V} \setminus \mathtt{Flipped}$.
   (b) Now for all nodes $w \in \mathcal{V} \setminus \{z\}$ flip a coin that comes out head with probability ρ, and if heads let $\widetilde{E_1} \coloneqq \widetilde{E_1} \cup \{(z, w)\}$. Further, if the coin comes out heads and $w \notin \mathtt{Flipped}$ then let $\widetilde{E_2} \coloneqq \widetilde{E_2} \cup \{\{z, w\}\}$.
   (c) Finally, let $\mathtt{Flipped} \coloneqq \mathtt{Flipped} \cup \{z\}$.

The above process ensures that $E_1 \sim \widetilde{E_1}$ as for any potential edge (z, w) an independent coin is flipped for whether or not to add an edge exactly once. Similarly, it also holds that $E_2 \sim \widetilde{E_2}$ as there is exactly one independent coin flip for each potential edge $\{z, w\}$.

What is left is thus to show that $\phi_{\mathrm{Diam}}(\widetilde{G_2}, \lambda) \implies \phi_{\mathrm{Dist}}(\widetilde{G_1}, v, \lambda)$. To see this let $z \in \mathcal{V}$, and let $\mathtt{dist}_{\widetilde{G_1}}$ and $\mathtt{dist}_{\widetilde{G_2}}$ denote the distance functions for the respective graphs. We now assume the LHS of the implication and prove the RHS. The LHS ensures that

$$\mathtt{dist}_{\widetilde{G_2}}(v, z) \leq \lambda. \tag{6}$$

This implies that there is a path of nodes with edges between them $w_1, \ldots, w_{\lambda'-1}, w_{\lambda'}$ for some $\lambda' \leq \lambda$ that connects v to z in $\widetilde{G_2}$ and where $z = w_{\lambda'}$. We now observe that any node in this path w_i is also at distance i from v in graph $\widetilde{G_2}$. To see this, observe that edges are added to $\widetilde{E_1}$ in an ordered fashion such that they do not change the distance to any nodes that had already their edges selected. This implies that any such node in this path, w_i, selected it edges before node w_{i+1}. Now, as edges being added to $\widetilde{E_2}$ are only added to nodes that have not yet had their edges selected, this implies that this path also exists in $\widetilde{G_1}$ which concludes the proof. □

Choosing a good emulation function. Let us now turn our attention to how to select a good emulation function. Before looking at a concrete function, let us consider what properties constitute a good emulation function. The only property of the emulation function that we have used so far is that all parties should emulate at least 1 node. [6] However, there are additional things that we want from a useful emulation function:

1. It should ensure a low distance from any sender in the graph resulting from the honest sending process.

2. The communication complexity of the protocol should be as small as possible.

Lemmas 2 and 3 bounds the probability that the honest sending process has a long distance in terms of the probability that an Erdős–Rényi graph with of size identical to the number of honest emulated nodes. Furthermore, looking ahead we will want to instantiate $\rho \approx \frac{\log(h_{\mathtt{E}})+\kappa}{h_{\mathtt{E}}}$ to obtain Erdős–Rényi graph that has a diameter logarithmic in $h_{\mathtt{E}}$ unless with a probability that is negligible in κ. Unfortunately, $h_{\mathtt{E}}$ will not be known at the time of instantiation, so therefore one will have to instantiate ρ with a lower bound on $h_{\mathtt{E}}$ in the denominator and similarly an upper bound in the denominator. For this discussion, let $h'_{\mathtt{E}}$ be such a lower bound and let us use $n_{\mathtt{E}}$ as an upper bound.

Now, note that the expected number of neighbours for a party is linear in ρ. To see this let $N \xleftarrow{\$} \mathsf{ER\text{-}Emulation}_p(\mathtt{E}, \rho)$. Now estimating the expected size using Bernoulli's inequality (Lemma 12) we get:

$$\begin{aligned} \mathrm{E}[|N|] &= \sum_{r \in \mathcal{P} \setminus \{p\}} 1 - (1-\rho)^{\mathtt{E}(p)\cdot\mathtt{E}(r)} \\ &\leq \sum_{r \in \mathcal{P} \setminus \{p\}} \rho \cdot \mathtt{E}(p) \cdot \mathtt{E}(r) \\ &\leq \rho \cdot \mathtt{E}(p) \cdot n_{\mathtt{E}}. \end{aligned} \tag{7}$$

Hence, for ρ chosen according to the above, a rough bound on the expected communication complexity will be $O\left((\log(n_{\mathtt{E}}) + \kappa) \cdot \frac{n_{\mathtt{E}}^2}{h'_{\mathtt{E}}}\right)$. A good emulation function is thus one which makes this value as small as possible. As our emulation function we choose

$$\mathtt{E}(p) \coloneqq \lceil \alpha_p \cdot n \rceil. \tag{8}$$

For this emulation function above we can derive the following bounds using only the honest weight assumption:

$$h_{\mathtt{E}} = \sum_{p \in \mathcal{H}} \mathtt{E}(p) \geq \sum_{p \in \mathcal{H}} \alpha_p \cdot n \geq \gamma \cdot n. \tag{9}$$

and

$$n_{\mathtt{E}} = \sum_{p \in \mathcal{P}} \mathtt{E}(p) \leq \sum_{p \in \mathcal{P}} (\alpha_p \cdot n + 1) = 2 \cdot n. \tag{10}$$

This results in an expected communication complexity that is upper bounded by $O\left((\log(n) + \kappa) \cdot \frac{n}{\gamma}\right)$ while obtaining a logarithmic diameter.

[6] This property was used in the proof of Lemma 2.

Proving security of our theoretical flooding protocol. We are now ready to prove that the probability that $\pi_{\text{Flood}}(\mathsf{ER\text{-}Emulation}(\mathtt{E}, \rho))$ protocol does not ensure timely delivery is negligible for certain choices of $\mathtt{E}$ and ρ. To prove this we make use of probabilistic bounds on the diameter of undirected Erdős–Rényi graphs from [MNT22] as well as Lemmas 1 to 3. We restate the graph result from [MNT22] and provide an instantiation with concrete values simplification in Appendix B.

As a first step, we bound the probability that the distance of the honest sending process using the neighbourhood selection algorithm $\mathsf{ER\text{-}Emulation}(\mathtt{E}, \rho)$ has a large distance from the sender.

Lemma 4. *Let* $\mathtt{E}(p) \coloneqq \lceil \alpha_p \cdot n \rceil$*, let* $d \in [7, \infty]$*, and let* $\rho \coloneqq \frac{d}{\gamma \cdot n}$*. Further, let* $p \in \mathcal{H}$ *and* $G \xleftarrow{\$} \mathit{HSP}(p, \mathit{ER\text{-}Emulation}(\mathtt{E}, \rho), ((7 \cdot \log\left(\frac{n}{d}\right) + 2))$*. Then*

$$\begin{aligned} &\Pr\left[\phi_{\mathit{Dist}}\left(G, p, \left(7 \cdot \log\left(\frac{n}{d}\right) + 2\right)\right)\right] \\ &\geq 1 - \left(2 \cdot n \cdot \left(e^{-\frac{d}{18}} + \left(6 \cdot \log\left(\frac{n}{d}\right) + 1\right) \cdot e^{-\frac{7 \cdot d}{108}}\right) + e^{-\gamma \cdot n \cdot \left(\frac{d}{9} - 2\right)}\right). \end{aligned} \tag{11}$$

Proof. As $h_{\mathtt{E}} \leq n_{\mathtt{E}}$, Equations (9) and (10) ensures that $h_{\mathtt{E}} \in [\gamma \cdot n, 2 \cdot n]$. Corollary 5 ensures that for $G' \xleftarrow{\$} \mathcal{G}_{\text{ER}}\left(h_{\mathtt{E}}, \frac{d}{h_{\mathtt{E}}}\right)$ we have

$$\begin{aligned} &\Pr\left[\phi_{\text{Diam}}\left(G', 7 \cdot \log\left(\frac{h_{\mathtt{E}}}{2 \cdot d}\right) + 2\right)\right] \\ &\geq 1 - \left(h_{\mathtt{E}} \cdot \left(e^{-\frac{d}{18}} + \left(6 \cdot \log\left(\frac{h_{\mathtt{E}}}{2 \cdot d}\right) + 1\right) \cdot e^{-\frac{7 \cdot d}{108}}\right) + e^{-h_{\mathtt{E}} \cdot \left(\frac{d}{9} - 2\right)}\right). \end{aligned} \tag{12}$$

The probability that $\phi_{\text{Diam}}\left(G', 7 \cdot \log\left(\frac{h_{\mathtt{E}}}{2 \cdot d}\right) + 2\right)$ holds monotonously increases when the edge probability increases and hence this probability also holds for $G' \xleftarrow{\$} \mathcal{G}_{\text{ER}}\left(h_{\mathtt{E}}, \frac{d}{\gamma \cdot n}\right)$. Further for any graph G and natural numbers $a, b \in \mathbb{N}$ s.t. $a \leq b$ we have that $\phi_{\text{Diam}}(G', a) \Rightarrow \phi_{\text{Diam}}(G', b)$. Hence, we have that $G' \xleftarrow{\$} \mathcal{G}_{\text{ER}}\left(h_{\mathtt{E}}, \frac{d}{\gamma \cdot n}\right)$ satisfies

$$\begin{aligned} &\Pr\left[\max_{h \in [\gamma \cdot n, 2 \cdot n]} \phi_{\text{Diam}}\left(G, 7 \cdot \log\left(\frac{h}{2 \cdot d}\right) + 2\right)\right] \\ &\geq 1 - \max_{h \in [\gamma \cdot n, 2 \cdot n]} \left(h \cdot \left(e^{-\frac{d}{18}} + \left(6 \cdot \log\left(\frac{h}{2 \cdot d}\right) + 1\right) \cdot e^{-\frac{7 \cdot d}{108}}\right) + e^{-h \cdot \left(\frac{d}{9} - 2\right)}\right). \end{aligned} \tag{13}$$

Now, applying Lemmas 2 and 3 as well as inserting bounds for the maximum value for the expressions we obtain Equation (11). □

As a direct corollary of Lemmas 1 and 4, we get that the probability that $\pi_{\text{Flood}}(\mathsf{ER\text{-}Emulation}(\mathtt{E}, \rho))$ ensures timely delivery is overwhelming when choosing $\mathtt{E}$ and ρ as discussed above.

Corollary 1. *Let* $\mathtt{E}(p) \coloneqq \lceil \alpha_p \cdot n \rceil$*, let* $d \in [7, \infty]$*, and let* $\rho \coloneqq \frac{d}{\gamma \cdot n}$*. If* m *is a message that is input to some honest party in either* $\pi_{\mathit{Flood}}(\mathit{ER\text{-}Emulation}(\mathtt{E}, \rho))$ *then*

$$\begin{aligned} &\Pr\left[\mathtt{Timely}_m\left(\left(7 \cdot \log\left(\frac{n}{d}\right) + 2\right) \cdot \delta_{\mathit{Channel}}\right)\right] \\ &\geq 1 - \left(2 \cdot n \cdot \left(e^{-\frac{d}{18}} + \left(6 \cdot \log\left(\frac{n}{d}\right) + 1\right) \cdot e^{-\frac{7 \cdot d}{108}}\right) + e^{-\gamma \cdot n \cdot \left(\frac{d}{9} - 2\right)}\right). \end{aligned} \tag{14}$$

3.4 A Practical Protocol

ER-Emulation is unfortunately not a practical neighbourhood selection algorithm, as it requires each party to do n coin-flips per message that is sent and forwarded. To rectify this we first introduce two intermediate algorithms (Fast-ER-Emulation and Practical-ER-Emulation) by doing gradual changes to ER-Emulation, until we finally arrive at the algorithm WFS which is both practical, simple, and similar to algorithms deployed in practice (except that this algorithm maintains its complexity even for weighted corruptions).

3.4.1 Intermediary Neighbourhood Selection Algorithms

We first introduce the algorithm Fast-ER-Emulation. This is distributed identically to ER-Emulation but slightly more practical. The algorithm exploits that another way of creating an Erdős–Rényi is to first decide how many edges each node should have using the binomial distribution and afterward select this number of edges at random among the emulated nodes.

Below we will abuse notation slightly and write $\mathtt{E}(P)$ to denote the set of emulated nodes for a set of parties $P \subseteq \mathcal{P}$ and an emulation function $\mathtt{E}$,

$$\mathtt{E}(P) \triangleq \{p_i \mid p \in P \wedge i \in \{1, 2, \ldots, \mathtt{E}(p)\}\}.^{7}$$

Algorithm Fast-ER-Emulation$_p(\mathtt{E}, \rho)$

1: Let $N \coloneqq \varnothing$.
2: **for** $i \coloneqq 0;\ i < \mathtt{E}(p);\ i\mathrel{+}\mathrel{+}$ **do**
3: Sample $k \xleftarrow{\$} \mathcal{B}\left(|\mathtt{E}(\mathcal{P} \setminus \{p\})|, \rho\right)$.
4: Let A be k nodes sampled from $\mathtt{E}(\mathcal{P} \setminus \{p\})$ without replacement.
5: Set $N \coloneqq N \cup \left\{p' \mid p'_j \in A \wedge j \in \mathbb{N}\right\}$.
6: **end for**
7: **return** N.

We now show that Fast-ER-Emulation is distributed identically to ER-Emulation.

Lemma 5. *Let $\rho \in [0, 1]$, let $\lambda \in \mathbb{N}$, let $p \in \mathcal{H}$, and let $\mathtt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$ be an emulation function. If $G_1 \xleftarrow{\$}$ HSP$(p,$ ER-Emulation$(\mathtt{E}, \rho), \lambda)$ and $G_2 \xleftarrow{\$}$ HSP$(p,$ Fast-ER-Emulation$(\mathtt{E}, \rho), \lambda)$ then $G_1 \sim G_2$.*

Proof. It is sufficient to prove that for any p' we have that for $N_1 \xleftarrow{\$}$ ER-Emulation$_{p'}(\mathtt{E}, \rho)$ and $N_2 \xleftarrow{\$}$ Fast-ER-Emulation$_{p'}(\mathtt{E}, \rho)$ then $N_1 \sim N_2$. Both neighbourhood selection algorithms works by letting a party be included in the neighbourhood with the same probability as if there would have been an edge between between any of the emulated nodes of the two parties. By Equation (3) edges appear with ρ for N_1. It is hence sufficient to show that sufficient to look at the probability that the probability that any of the emulated nodes of p' select a node not belonging to p' with mutually independent probability ρ.

Let us look at a node $v \in \mathtt{E}(\{p'\})$ and calculate the probability that there are edges to a specific set of other nodes $U \subseteq \mathtt{E}(\mathcal{P} \setminus \{p'\})$ from v. Let A be the set of nodes v chooses. We now want to show that:

$$\Pr\left[U \subseteq A\right] = \rho^{|U|}. \tag{15}$$

[7] This set may be different from the acutal set of nodes that will be emulated in an execution of the protocol as dishonest parties might choose to deviate from the protocol. However, it is still useful to define the set in order to define honest behavior.

We let $\eta \coloneqq |\mathtt{E}(\mathcal{P} \setminus \{p'\})|$ and now apply the law of total probabilities for conditional events to obtain

$$\begin{aligned}\Pr[U \subseteq A] &= \sum_{i=0}^{\eta} \Pr[U \subseteq A \mid |A| = i] \cdot \Pr[|A| = i] \\ &= \sum_{i=|U|}^{\eta} \Pr[U \subseteq A \mid |A| = i] \cdot \Pr[|A| = i]\end{aligned} \tag{16}$$

For any $i \geq |U|$ we have that A is chosen uniformly among sets of size i. Of those sets exactly $\binom{\eta-|U|}{i-|U|}$ includes U. Hence,

$$\Pr[U \subseteq A \mid |A| = i] = \frac{\binom{\eta-|U|}{i-|U|}}{\binom{\eta}{i}}. \tag{17}$$

Inserting this we obtain

$$\begin{aligned}\Pr[U \subseteq A] &= \sum_{i=|U|}^{\eta} \frac{\binom{\eta-|U|}{i-|U|}}{\binom{\eta}{i}} \binom{\eta}{i} \cdot \rho^i \cdot (1-\rho)^{\eta-i} \\ &= \sum_{i=|U|}^{\eta} \binom{\eta-|U|}{i-|U|} \cdot \rho^i \cdot (1-\rho)^{\eta-i}.\end{aligned} \tag{18}$$

We now change the variable letting $r \coloneqq i + |U|$, factor out $\rho^{|U|}$, and use the binomial formula (Lemma 11) to obtain

$$\begin{aligned}\Pr[U \subseteq A] &= \sum_{r=0}^{\eta-|U|} \binom{\eta-|U|}{r} \cdot \rho^{r+|U|} \cdot (1-\rho)^{\eta-(r+|U|)} \\ &= \rho^{|U|} \cdot \sum_{r=0}^{\eta-|U|} \binom{\eta-|U|}{r} \cdot \rho^{r+|U|} \cdot (1-\rho)^{\eta-|U|-r} \\ &= \rho^{|U|} \cdot (\rho + (1-\rho))^{\eta-|U|} \\ &= \rho^{|U|}.\end{aligned} \tag{19}$$

Therefore we can conclude that all edges between emulated nodes appear with a mutually independent probability ρ and hence $G_1 \sim G_2$. □

A problem of Fast-ER-Emulation is that each party p needs to make $\mathtt{E}(p)$ number of draws from the binomial distribution. One way to avoid this is to make a single random draw for the number of nodes all emulated nodes should send to and then afterward choose this number of nodes uniformly without replacement. Below we present the algorithm Practical-ER-Emulation, which does exactly that.

Algorithm Practical-ER-Emulation$_p(\mathtt{E}, \rho)$

1: Let $N \coloneqq \varnothing$.
2: Sample $k \xleftarrow{\$} \mathcal{B}\left(\mathtt{E}(p) \cdot |\mathtt{E}(\mathcal{P} \setminus \{p\})|, \rho\right)$.
3: Let A be k nodes sampled from $\mathtt{E}(\mathcal{P} \setminus \{p\})$ without replacement.
4: Set $N \coloneqq \{p \mid p_i \in A \wedge i \in \mathbb{N}\}$.
5: **return** N.

Practical-ER-Emulation is not distributed identically to Fast-ER-Emulation, as there is a smaller expected overlap between the selected emulated nodes. However, it still holds that the graph resulting from the honest sending process based upon Practical-ER-Emulation has a higher chance of having a low distance from the sender than the graph resulting from the honest sending process based upon Fast-ER-Emulation. We make this intuition formal in the lemma below.

Lemma 6. *Let $\rho \in [0,1]$, let $\lambda \in \mathbb{N}$, let $p \in \mathcal{H}$, and let $\mathtt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$ be an emulation function. If $G_1 \xleftarrow{\$} \mathsf{HSP}(p, \mathsf{Fast\text{-}ER\text{-}Emulation}(\mathtt{E}, \rho), \lambda)$ and $G_2 \xleftarrow{\$} \mathsf{HSP}(p, \mathsf{Practical\text{-}ER\text{-}Emulation}(\mathtt{E}, \rho), \lambda)$ then*

$$\Pr[\phi_{Dist}(G_1, p, \lambda)] \leq \Pr[\phi_{Dist}(G_2, p, \lambda)]. \tag{20}$$

Proof. Let $G_1 = (\mathcal{H}, E_1)$ and $G_2 = (\mathcal{H}, E_2)$. To show the lemma we define a coupling $\widetilde{G_1} = (\mathcal{H}, \widetilde{E_1})$ and $\widetilde{G_2} = (\mathcal{H}, \widetilde{E_2})$ s.t. $G_1 \sim \widetilde{G_1}$ and $G_2 \sim \widetilde{G_2}$ and $\phi_{\mathrm{Dist}}(\widetilde{G_1}, p, \lambda) \implies \phi_{\mathrm{Dist}}(\widetilde{G_2}, p, \lambda)$. To show the implication it is sufficient to show that $\widetilde{E_1} \subseteq \widetilde{E_2}$. We define the coupling by sampling $\widetilde{G_1}$ and $\widetilde{G_2}$ in parallel with a defined a coupling between the neighbourhood selection algorithms for any party $p' \in \mathcal{H}$. We let $N_1 \xleftarrow{\$} \mathsf{Fast\text{-}ER\text{-}Emulation}_{p'}(\mathtt{E}, \rho)$ and $N_2 \xleftarrow{\$} \mathsf{Practical\text{-}ER\text{-}Emulation}_{p'}(\mathtt{E}, \rho)$. Again we define a coupling $\widetilde{N_1} \sim N_1$ and $\widetilde{N_2} \sim N_2$ and show that $\widetilde{N_1} \subseteq \widetilde{N_2}$. This is sufficient to ensure that $\widetilde{E_1} \subseteq \widetilde{E_2}$ because this ensures that all parties that selects their neighbours when constructing $\widetilde{G_1}$ also gets to select their neighbours in the construction of $\widetilde{G_2}$.

We define $\widetilde{N_1}$ and $\widetilde{N_2}$ via the following process.

1. Initially, let $\widetilde{N_1} = \widetilde{N_2} \coloneqq \varnothing$.
2. We now initialize variables corresponding to the variables used in Line 3 of $\mathsf{Fast\text{-}ER\text{-}Emulation}_{p'}(\mathtt{E}, \rho)$ by sampling $\widetilde{k_1^1} \xleftarrow{\$} \mathcal{B}\left(|\mathtt{E}(\mathcal{P} \setminus \{p\})|, \rho\right), \ldots, \widetilde{k_1^{\mathtt{E}(p')}} \xleftarrow{\$} \mathcal{B}\left(|\mathtt{E}(\mathcal{P} \setminus \{p\})|, \rho\right)$, and a variable corresponding $\widetilde{k_2} \coloneqq \sum_{i=1}^{\mathtt{E}(p')} \widetilde{k_1^i}$ to the k in Line 2 of $\mathsf{Practical\text{-}ER\text{-}Emulation}_{p'}(\mathtt{E}, \rho)$
3. Additionally, we initialize $\widetilde{A_1^1} \coloneqq \varnothing, \ldots, \widetilde{A_1^{\mathtt{E}(p')}} \coloneqq \varnothing$ that corresponds to the variable A in Line 4 of $\mathsf{Fast\text{-}ER\text{-}Emulation}_{p'}(\mathtt{E}, \rho)$ and $\widetilde{A_2} \coloneqq \varnothing$ that corresponds to the variable A in Line 3 of $\mathsf{Practical\text{-}ER\text{-}Emulation}_{p'}(\mathtt{E}, \rho)$.
4. Now for $i \in \{1, \ldots, \mathtt{E}(p')\}$ and for $j \in \left\{1, \ldots, \widetilde{k_1^i}\right\}$ do:
   (a) Set $\mathtt{Accepted}_1 \coloneqq \bot$ and $\mathtt{Accepted}_2 \coloneqq \bot$.
   (b) While $\neg\mathtt{Accepted}_1 \vee \neg\mathtt{Accepted}_2$ do:
      i. Sample v uniformly in $\mathtt{E}(\mathcal{P} \setminus \{p'\})$.
      ii. If $v \notin \widetilde{A_2} \wedge \neg\mathtt{Accepted}_2$ set $\widetilde{A_2} \coloneqq \widetilde{A_2} \cup \{v\}$ and $\mathtt{Accepted}_2 \coloneqq \top$.
      iii. If $v \notin \widetilde{A_1^i} \wedge \neg\mathtt{Accepted}_1$ set $\widetilde{A_1^i} \coloneqq \widetilde{A_1^i} \cup \{v\}$ and $\mathtt{Accepted}_1 \coloneqq \top$.
5. Finally, set $\widetilde{A_1} \coloneqq \bigcup_{i=1}^{\mathtt{E}(p')} \widetilde{A_1^i}$, set$\widetilde{N_1} \coloneqq \left\{p \mid p_i \in \widetilde{A_1} \wedge i \in \mathbb{N}\right\}$ and set $\widetilde{N_2} \coloneqq \left\{p \mid p_i \in \widetilde{A_2} \wedge i \in \mathbb{N}\right\}$.

Note at first that $\widetilde{N_1} \sim N_1$ as the procedure simply uses rejection sampling to sample $\widetilde{k_1^i}$ elements from $\mathtt{E}(\mathcal{P} \setminus \{p'\})$ without repetition and adds these to $\widetilde{A_1}$, and hence have the same distribution as $\mathsf{Fast\text{-}ER\text{-}Emulation}_{p'}(\mathtt{E}, \rho)$. Furthermore, note that the procedure simply samples $\widetilde{k_2} = \sum_{i=1}^{\mathtt{E}(p')} \widetilde{k_1^i}$ elements from $\mathtt{E}(\mathcal{P} \setminus \{p'\})$ without repetition via. rejection sampling and add these to $\widetilde{A_2}$. Because $\widetilde{k_2}$ is the sum of $\mathtt{E}(p')$ independent random variables that are each distributed as $\mathcal{B}\left(|\mathtt{E}(\mathcal{P} \setminus \{p'\})|, \rho\right)$ we get that $\widetilde{k_2} \sim \mathcal{B}\left(\mathtt{E}(p') \cdot |\mathtt{E}(\mathcal{P} \setminus \{p'\})|, \rho\right)$ and hence that $\widetilde{N_2} \sim N_2$.

Furthermore, the algorithm ensures that $\widetilde{N_1} \subseteq \widetilde{N_2}$ because at any point in the algorithm we have that $\widetilde{A_2}$ is a superset of any $\widetilde{A_1^i}$. □

As a simple corollary of Lemmas 1 and 4 to 6 we get that the probability that $\pi_{\text{Flood}}(\mathsf{Fast\text{-}ER\text{-}Emulation}(\mathtt{E}, \rho))$ and $\pi_{\text{Flood}}(\mathsf{Practical\text{-}ER\text{-}Emulation}(\mathtt{E}, \rho))$ do not ensure timely delivery is negligible for a certain choice of $\mathtt{E}$ and ρ.

Corollary 2. *Let* $\mathtt{E}(p) \coloneqq \lceil \alpha_p \cdot n \rceil$*, let* $d \in [7, \infty]$*, and let* $\rho \coloneqq \frac{d}{\gamma \cdot n}$*. If* m *is a message that is input to some honest party in either*

- $\pi_{Flood}(\mathsf{Fast\text{-}ER\text{-}Emulation}(\mathtt{E}, \rho))$,
- *or* $\pi_{Flood}(\mathsf{Practical\text{-}ER\text{-}Emulation}(\mathtt{E}, \rho))$

then

$$\begin{aligned} &\Pr\left[\mathtt{Timely}_m\left(\left(7 \cdot \log\left(\frac{n}{d}\right) + 2\right) \cdot \delta_{Channel}\right)\right] \\ &\geq 1 - \left(2 \cdot n \cdot \left(e^{-\frac{d}{18}} + \left(6 \cdot \log\left(\frac{n}{d}\right) + 1\right) \cdot e^{-\frac{7 \cdot d}{108}}\right) + e^{-\gamma \cdot n \cdot \left(\frac{d}{9} - 2\right)}\right). \end{aligned} \tag{21}$$

3.4.2 A Practical Neighbourhood Selection Algorithm

Neither Fast-ER-Emulation nor Practical-ER-Emulation is close to algorithms already deployed in practice where each party forwards a message to a *fixed* number of parties. We incorporate this design paradigm in the algorithm $\mathsf{WFS}(\mathtt{E}, k)$ (abbreviation for "Weighted Fan-out Selection") which aside from an emulation function $\mathtt{E}$ also takes a security parameter k.

The idea of the very simple algorithm is that each party p chooses $K = k \cdot \mathtt{E}(p)$, for $\mathtt{E}(p) = \lceil \alpha_p \cdot n \rceil$, number of neighbours (excluding himself). The neighbours are chosen according to weighted sampling without replacement [BHPS16]. More precisely, p chooses K neighbours, and the probability to choose the tuple of neighbours $(q_1, \ldots, q_K)$ is defined as follows:

$$\Pr\left((q_1, \ldots, q_K)\right) = \prod_{i=1}^{K} \frac{\mathtt{E}(q_i)}{n_{\mathtt{E}} - \mathtt{E}(p) - \mathtt{E}(q_1) - \cdots - \mathtt{E}(q_{i-1})}.$$

The probability to choose a certain neighborhood set $\{q_1, \ldots, q_K\}$ is then the sum over the probabilities over all the permuted tuples. We denote by $\mathcal{D}_K$ the resulting distribution.

Algorithm $\mathsf{WFS}_p(\mathtt{E}, k)$

1: Let $N \coloneqq \varnothing$.
2: Set $K \coloneqq k \cdot \mathtt{E}(p)$.
3: Sample $A \xleftarrow{\$} \mathcal{D}_K$ parties from $\mathcal{P} \setminus \{p\}$, and add these to N.
4: **return** N.

We now relate the probability that graph constructed by the honest sending process of Practical-ER-Emulation has a low distance from the sender to the probability that the honest sending process of WFS has a low distance from the sender.

Lemma 7. *Let* $\rho \in [0, 1]$*, let* $\epsilon \in [0, 1]$ *let* $\lambda \in \mathbb{N}$*, let* $p \in \mathcal{H}$*, let* $k \geq \lceil (1 + \epsilon) \cdot n_{\mathtt{E}} \cdot \rho \rceil$*, and let* $\mathtt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$ *be an emulation function. If* $G_1 \xleftarrow{\$} \mathsf{HSP}(p, \mathsf{Practical\text{-}ER\text{-}Emulation}(\mathtt{E}, \rho), \lambda)$ *and* $G_2 \xleftarrow{\$} \mathsf{HSP}(p, \mathsf{WFS}(\mathtt{E}, k), \lambda)$ *then*

$$\Pr[\phi_{Dist}(G_1, p, \lambda)] - |\mathcal{H}| \cdot e^{-\frac{\epsilon^2 \cdot (n-1) \cdot \rho}{3}} \leq \Pr[\phi_{Dist}(G_2, p, \lambda)]. \tag{22}$$

Proof. Let $G_1 = (\mathcal{H}, E_1)$ and $G_2 = (\mathcal{H}, E_2)$. To show the lemma we define again a coupling $\widetilde{G_1} = (\mathcal{H}, \widetilde{E_1})$ and $\widetilde{G_2} = (\mathcal{H}, \widetilde{E_2})$ s.t. $G_1 \sim \widetilde{G_1}$ and $G_2 \sim \widetilde{G_2}$. For each party $p' \in \mathcal{H}$ we introduce a random variable $X_{p'}$ which denotes the number of outgoing edges this party has in $\widetilde{G_1}$. We further define C to be the event $\bigcap_{p' \in \mathcal{H}}(X_{p'} \leq (1+\epsilon) \cdot \mathtt{E}(p') \cdot |\mathtt{E}(\mathcal{P} \setminus \{p'\})| \cdot \rho)$ (this event can be thought of as that no party picks "outrageously many neighbours"). We now wish to show two things:

$$\phi_{\text{Dist}}(\widetilde{G_1}, p, \lambda) \wedge C \implies \phi_{\text{Dist}}(\widetilde{G_2}, p, \lambda), \tag{23}$$

and

$$\Pr[\phi_{\text{Dist}}(\widetilde{G_1}, p, \lambda) \wedge C] \geq \Pr[\phi_{\text{Dist}}(G_1, p, \lambda)] - |\mathcal{H}| \cdot e^{-\frac{\epsilon^2 \cdot (n-1) \cdot \rho}{3}}. \tag{24}$$

To show Equation (23) it is sufficient to show that $C \implies \widetilde{E_1} \subseteq \widetilde{E_2}$. As in the proof of Lemma 6 we define the coupling by sampling $\widetilde{G_1}$ and $\widetilde{G_2}$ in parallel with a defined a coupling between the neighbourhood selection algorithms for any party $p' \in \mathcal{H}$. We let $N_1 \xleftarrow{\$} \mathsf{Practical\text{-}ER\text{-}Emulation}_{p'}(\mathtt{E}, \rho)$ and $N_2 \xleftarrow{\$} \mathsf{WFS}_{p'}(\mathtt{E}, k)$. Again we define a coupling $\widetilde{N_1} \sim N_1$ and $\widetilde{N_2} \sim N_2$ and show that $C \implies \widetilde{N_1} \subseteq \widetilde{N_2}$ which again will imply that $C \implies \widetilde{E_1} \subseteq \widetilde{E_2}$. Again we define a coupling $\widetilde{N_1} \sim N_1$ and $\widetilde{N_2} \sim N_2$ and show that $\widetilde{N_1} \subseteq \widetilde{N_2}$. This is sufficient to ensure that $\widetilde{E_1} \subseteq \widetilde{E_2}$ because this ensures that all parties that selects their neighbours when constructing $\widetilde{G_1}$ also gets to select their neighbours in the construction of $\widetilde{G_2}$.

We define $\widetilde{N_1}$ and $\widetilde{N_2}$ via the following process.

1. Initially, let $\widetilde{N_1} = \widetilde{N_2} \coloneqq \varnothing$.
2. We further initialize variables corresponding to the variables k and A used in Lines 2 and 3 of $\mathsf{Practical\text{-}ER\text{-}Emulation}_{p'}(\mathtt{E}, \rho)$ by sampling $\widetilde{k_1} \xleftarrow{\$} \mathcal{B}(\mathtt{E}(p') \cdot |\mathtt{E}(\mathcal{P} \setminus \{p'\})|, \rho)$ and set $\widetilde{A_1} \coloneqq \varnothing$. We also initialize a variable $\widetilde{K_2} \coloneqq \mathtt{E}(p') \cdot k$ identical to the variable K in Line 2 of $\mathsf{WFS}_{p'}(\mathtt{E}, k)$.
3. Now for $i \in \left\{1, \ldots, \max(\widetilde{k_1}, \widetilde{K_2})\right\}$ do:
   (a) Set $\mathtt{Accepted}_1 \coloneqq i > \widetilde{k_1}$ and $\mathtt{Accepted}_2 \coloneqq i > \widetilde{K_2}$.
   (b) While $\neg\mathtt{Accepted}_1 \vee \neg\mathtt{Accepted}_2$ do:
      i. Sample p_j uniformly in $\mathtt{E}(\mathcal{P} \setminus \{p'\})$.
      ii. If $p \notin \widetilde{N_2} \wedge \neg\mathtt{Accepted}_2$ set $\widetilde{N_2} \coloneqq \widetilde{N_2} \cup \{p\}$ and $\mathtt{Accepted}_2 \coloneqq \top$. [8]
      iii. If $v \notin \widetilde{A_1} \wedge \neg\mathtt{Accepted}_1$ set $\widetilde{A_1} \coloneqq \widetilde{A_1} \cup \{v\}$ and $\mathtt{Accepted}_1 \coloneqq \top$.
4. Finally, set $\widetilde{N_1} \coloneqq \left\{p \mid p_i \in \widetilde{A_1} \wedge i \in \mathbb{N}\right\}$.

Note at first that $\widetilde{N_1} \sim N_1$ as the procedure simply uses rejection sampling to sample $\widetilde{k_1}$ elements from $\mathtt{E}(\mathcal{P} \setminus \{p'\})$ without repetition and adds these to $\widetilde{A_1}$, and hence have the same distribution as $\mathsf{Practical\text{-}ER\text{-}Emulation}_{p'}(\mathtt{E}, \rho)$. Similarly, it is clear that $\widetilde{N_2} \sim N_2$ as once again rejection sampling is used to pick $\mathtt{E}(p') \cdot k$ from $\mathcal{P} \setminus \{p'\}$ weighted according to $\mathtt{E}$. It is also immediate to see that $C \implies \widetilde{N_1} \subseteq \widetilde{N_2}$ as if C happens then surely the max of $\widetilde{k_1}$ and $\widetilde{K_2}$ is $\widetilde{K_2}$, as $n_{\mathtt{E}} > |\mathtt{E}(\mathcal{P} \setminus \{p'\})|$. Hence all parties that will be added to $\widetilde{N_1}$ will also be added to $\widetilde{N_2}$.

[8] Note that p in this step does not refer to the sender in the honest sending process but rather the party that is supposed to emulate node p_j.

It is thus left to show Equation (24). We have that

$$\Pr[\phi_{\text{Dist}}(\widetilde{G_1}, p, \lambda) \wedge C] = \Pr[\phi_{\text{Dist}}(\widetilde{G_1}, p, \lambda)] - \Pr[\neg C]. \tag{25}$$

As we have already argued that $G_1 \sim \widetilde{G_1}$ it is sufficient to show that

$$\Pr[\neg C] \leq |\mathcal{H}| \cdot e^{-\frac{\epsilon^2 \cdot (n-1) \cdot \rho}{3}}. \tag{26}$$

Further, using a union bound we have that.

$$\begin{aligned} \Pr[\neg C] &= \Pr\left[\bigcup_{p' \in \mathcal{H}} (X_{p'} > (1+\epsilon) \cdot \mathtt{E}(p') \cdot |\mathtt{E}(\mathcal{P} \setminus \{p'\})| \cdot \rho)\right] \\ &\leq \sum_{p' \in \mathcal{H}} \Pr\left[X_{p'} > (1+\epsilon) \cdot \mathtt{E}(p') \cdot |\mathtt{E}(\mathcal{P} \setminus \{p'\})| \cdot \rho\right]. \end{aligned} \tag{27}$$

Furthermore for each party it is clear that the number of neighbours selected is less than the selected emulated neighbours. We let $Y_{p'}$ denote this number of emulated neighbours. Hence, it is sufficient to show that for any $p' \in \mathcal{H}$ we have,

$$\Pr\left[Y_{p'} > (1+\epsilon) \cdot \mathtt{E}(p') \cdot |\mathtt{E}(\mathcal{P} \setminus \{p'\})| \cdot \rho\right] \leq e^{-\frac{\epsilon^2 \cdot (n-1) \cdot \rho}{3}}. \tag{28}$$

Now, as for any $p'' \in \mathcal{P}$ we have that $\mathtt{E}(p'') \geq 1$ it follows that $|\mathtt{E}(\mathcal{P} \setminus \{p'\})| \geq n-1$. This makes desired equation follows from the Chernoff bound (Lemma 14)

$$\begin{aligned} \Pr\left[Y_{p'} > (1+\epsilon) \cdot \mathtt{E}(p') \cdot |\mathtt{E}(\mathcal{P} \setminus \{p'\})| \cdot \rho\right] &\leq e^{-\frac{\epsilon^2 \cdot |\mathtt{E}(\mathcal{P} \setminus \{p'\})| \cdot \mathtt{E}(p') \cdot \rho}{3}} \\ &\leq e^{-\frac{\epsilon^2 \cdot (n-1) \cdot \rho}{3}}. \end{aligned} \tag{29}$$

□

Our final protocol is the protocol obtained by instantiating the flooding skeleton π_{Flood} with the neighbourhood selection algorithm WFS that again is to be instantiated with the imitation function discussed above. We name this protocol the *weighted fan-out flooding* protocol and use the abbreviation $\mathsf{WFF}(k) \coloneqq \pi_{\text{Flood}}(\mathsf{WFS}(\mathtt{E}, k))$ for $\mathtt{E}(p) \coloneqq \lceil \alpha_p \cdot n \rceil$.

We now provide a corollary that bounds the concrete probability that a message that is input via. WFF delivered timely.

Corollary 3. *Let $k \in \mathbb{N}$ s.t. $k \geq \frac{42}{\gamma}$. If m is a message that is input to some honest party in $\mathsf{WFF}(k)$ then*

$$\begin{aligned} &\Pr\left[\mathtt{Timely}_m\left(\left(7 \cdot \log\left(\frac{n \cdot 6}{k \cdot \gamma}\right) + 2\right) \cdot \delta_{Channel}\right)\right] \\ &\geq 1 - \left(2 \cdot n \cdot \left(e^{-\frac{k \cdot \gamma}{108}} + \left(6 \cdot \log\left(\frac{n \cdot 6}{k \cdot \gamma}\right) + 1\right) \cdot e^{-\frac{7 \cdot k \cdot \gamma}{648}}\right) + e^{-\gamma \cdot n \cdot \left(\frac{k \cdot \gamma}{54} - 2\right)} + n \cdot e^{-\frac{(n-1) \cdot k}{n \cdot 24}}\right). \end{aligned} \tag{30}$$

Proof. Let $\mathtt{E}(p) \coloneqq \lceil \alpha_p \cdot n \rceil$. Lemma 1 ensures that is enough to reason about the honest sending process with the neighbourhood selection algorithm $\mathsf{WFS}(\mathtt{E}, k)$. We observe that

$$n_{\mathtt{E}} = \sum_{p \in \mathcal{P}} \lceil \alpha_p \cdot n \rceil \leq \sum_{p \in \mathcal{P}} \alpha_p \cdot n + 1 = 2 \cdot n. \tag{31}$$

We let $d \coloneqq \frac{k \cdot \gamma}{6}$, and note that the precondition for Lemma 7 is satisfied for $\rho \coloneqq \frac{d}{\gamma \cdot n}$ so we instantiate this letting $\epsilon \coloneqq \frac{1}{2}$, and using the bound above on $n_{\mathtt{E}}$. Furthermore, note that $|\mathcal{H}| \leq n$ and hence it is sufficient to bound the probability that the honest sending process of $\mathsf{Practical\text{-}ER\text{-}Emulation}(\mathtt{E}, \rho)$ has a large distance from the sender. Equation (30) now follows by Lemmas 4 to 6. □

As observed earlier, it is sufficient to bound the probability that a message is timely delivered in order to bound the probability that any of the two properties of a flooding protocol is achieved. We conclude with the following theorem that states that WFF is a flooding protocol.

Theorem 2. *Let* $k := \frac{\log(n)+\kappa}{\gamma}$. *WFF*$(k)$ *is a* Δ*-flooding protocol for* $\Delta := \left(7 \cdot \log\left(\frac{n \cdot 6}{k \cdot \gamma}\right) + 2\right) \cdot \delta_{Channel}$ *and has communication complexity less than* $k \cdot 2 \cdot n$.

4 Asymptotic Optimality and Practical Considerations

Our results from Section 3.4.2 show that the protocol WFF(k) provides provably secure flooding. With respect to efficiency, the results show that there are two possible drawbacks: First, the emulation function $\mathtt{E}(p) = \lceil \alpha_p \cdot n \rceil$ forces parties with very high weight to send to many parties, which lead to bandwidth issues. Secondly, Theorem 2 shows that in our protocol, the number of parties each node has to send to increases logarithmically in the total number of nodes. In this section, we show that both properties are inherent for "flooding protocols".

4.1 Workload of Heavy Parties

It is easy to see that in at least in extreme cases, very heavy parties have to send to a lot of other parties: If there is a single party that has the majority of the total weight, it could be that only this party and an additional one are honest. Since the heavy party is the only one that can be relied upon for message delivery, it needs to send to all other parties. The following lemma generalizes this idea to less extreme settings.

Lemma 8. *For any protocol* Π *that guarantees delivery to* all honest *parties, and for any subset* $S \subseteq \mathcal{P}$ *such that* $\sum_{p \in S} \alpha_p \geq \gamma$, *we have with overwhelming probability that*

$$\sum_{p \in S} \mathtt{degree}_{\Pi}(p) \geq |\mathcal{P} \setminus S|. \tag{32}$$

Proof. Let S be any such set. By the honesty assumption it could be that there is exactly one honest party in $\mathcal{P} \setminus S$. To guarantee delivery to this party, some party in S must send to it. Since it cannot be distinguished which party in $\mathcal{P} \setminus S$ is honest, the parties in S must send to all parties in $\mathcal{P} \setminus S$. □

Another consequence of Lemma 8 is that having a huge number of nodes with very little weight also increases the workload for all other nodes, as we show below.

Corollary 4 (Implications of Lemma 8 for weight distributions with tiny weights)**.** *Assume there is a large set* $T \subseteq \mathcal{P}$ *of parties with combined relative weight* $\leq 1 - \gamma$ *and* $|T| \geq n - \epsilon$ *for some constant* $\epsilon > 0$, *and define* $S := \mathcal{P} \setminus T$. *Then, the average degree of the parties in* S *must be at least* $\frac{n-\epsilon}{\epsilon} \in \Omega(n)$ *with overwhelming probability.*

Proof. Since $\sum_{p \in S} \alpha_p = 1 - \sum_{p \in T} \alpha_p \geq \gamma$, Lemma 8 implies that the average degree of the parties in S is at least $\frac{|\mathcal{P} \setminus S|}{|S|}$ with overwhelming probability. By assumption, we have $\frac{|\mathcal{P} \setminus S|}{|S|} = \frac{|T|}{n-|T|} \geq \frac{n-\epsilon}{\epsilon} \in \Omega(n)$. □

Limiting the workload. As we have seen above, having very heavy or many very light parties necessarily yields a large number of outgoing connections for some of the nodes. This is not only undesirable but may also become prohibitive in practice due to limited network bandwidth. If the flooding is deployed, say for a proof-of-stake blockchain, this can be mitigated by putting a lower and an upper limit on the amount of stake for actively participating nodes. This implies that people holding a lot of stake need to split their stake over several nodes (which is anyway beneficial for decentralization if they are run in different locations), and people with too little stake need to, e.g., delegate their stake to another node if supported by the blockchain. The latter can still passively participate by fetching data from other nodes as discussed for zero-weight parties in Section 5.2.

4.2 Message Complexity Grows Logarithmically in Number of Parties

It is well known that Erdős–Rényi graphs are connected with high probability if and only if edges are included with probability larger than $\frac{\log n}{n}$ [Bol01, Theorem 7.3]. This means the expected degree of a node must be larger than $\log n$ to obtain a connected graph, even without considering corruptions. Since our proofs in Section 3 depart from Erdős–Rényi graphs, one cannot hope to prove a better communication complexity with our proof techniques.

On the other hand, our final protocol $\mathsf{WFF}(k)$ does not choose neighbors in the way Erdős–Rényi graphs are constructed, but more closely correspond to so-called directed k-out graphs, which have also been considered in the literature. Those are directed graphs where for each node v independently, k uniformly random other nodes are sampled and directed edges from v to the k sampled nodes are added. It is known that such graphs are connected with probability approaching 1 for $n \to \infty$ already for constant $k = 2$ [FF82]. Hence, at least without corruptions, $O(n)$ overall communication complexity should be enough for our protocol. When considering corruptions, however, a result by Yagan and Makowski [YM13] implies that $\log n$ connections for each node are necessary, as we show below. This shows that $\mathsf{WFF}(k)$ and Theorem 2 are asymptotically optimal, at least for the special case in which all parties have the same weight.

Lemma 9. *For any flooding protocol in which all honest parties send to k uniformly chosen nodes and delivery to all honest nodes is guaranteed with probability $\geq 1/2$ where up to a $(1-\gamma)$ fraction of nodes can be corrupted, we have for sufficiently large n that*

$$k \geq \frac{\log n}{\gamma + 1/n - \log(1 - \gamma - 1/n)}.$$

Proof. Yagan and Makowski [YM13] have considered the setting in which for each of the n nodes p_i, k distinct random other nodes are sampled and undirected edges between p_i and all k sampled nodes are added to a graph. They then consider the subgraph H consisting of the first $\lfloor \gamma' n \rfloor$ nodes for some constant $\gamma' \in (0, 1)$ and show in [YM13, Theorem 3.2] that

$$k < \frac{\log n}{\gamma' - \log(1 - \gamma')} \implies \lim_{n \to \infty} \Pr[H \text{ contains isolated node}] = 1.$$

To translate this to our setting, first note that corrupting at most $\lfloor (1-\gamma)n \rfloor$ nodes from the end to leave the first $\lfloor \gamma n + 1 \rfloor$ parties honest is a valid adversarial strategy. To be compatible with the result above, we can set $\gamma' := \gamma + 1/n$. Further note that a node p being isolated in H has the same probability as an honest node not sending to any other honest node and no honest node sending to that one in a flooding protocol. In that case, if p is the sender in the flooding protocol, no honest node will receive the message, and if some other node is the sender, p will not receive the message. Hence, the flooding protocol will fail to deliver the message to all honest nodes in both cases. This implies that, for sufficiently large n, flooding protocols with $k < \frac{\log n}{\gamma + 1/n - \log(1-\gamma-1/n)}$ fail to deliver messages with high probability. □

5 Delivery to Parties With Zero Weight

So far, we have excluded parties with zero weight from participating in our protocol. While they are not relevant for the security of consensus protocols running on top of the network, it is still important in practice to allow such passive nodes to obtain the data from the blockchain, e.g., for connecting wallets. We discuss here some options that allow zero-weight parties to obtain data with advantages and disadvantages of the different approaches.

5.1 Adjusting the Emulation Function

Recall from Section 3.3 that we use the emulation function

$$\mathtt{E}(p) = \lceil \alpha_p \cdot n \rceil.$$

This implies that parties with weight 0, i.e., $\alpha_p = 0$ emulate 0 parties and consequently do not send anything and also do not receive anything. If we want to guarantee delivery to parties with zero weight, we can instead use the emulation function

$$\mathtt{E}'(p) \coloneqq \lfloor \alpha_p \cdot n \rfloor + 1.$$

This ensures that all parties emulate at least one party. Furthermore, inequalities (9) and (10) from Section 3.3 also hold for $\mathtt{E}'$ and our results from that section follow similarly as for $\mathtt{E}$. Hence, using the emulation function $\mathtt{E}'$ guarantees delivery to all parties, includes those with weight 0.

A downside of this approach is that considering parties with weight 0 opens up the system for Sybil attacks: An attacker can easily add additional zero-weight nodes to the system and thereby increase n arbitrarily without changing any α_p. According to $\mathtt{E}'$, the work required from honest parties with nonzero-weight thus increases linearly in n, allowing the attacker to increase the workload of honest parties arbitrarily. Such approach therefore is only practical if there is some mechanism for preventing Sybil attacks in place.

5.2 Fetching Data

Since guaranteeing that all zero-weight parties receive all data in the flooding process can substantially increase the workload for honest parties, we here provide an alternative. The idea is to exclude zero-weight parties from the regular protocol as we do in our main results, and to allow those parties to obtain the state by querying other nodes. To prevent Sybil attacks, parties with non-zero weight can then refuse to answer if they receive too many requests. This ensures that the flooding among parties with non-zero weight, which is critical for consensus of the blockchain, cannot be negatively affected by zero-weight parties; the worst outcome of Sybil attacks is that honest zero-weight parties cannot obtain data from the blockchain.

We formalize this idea in the algorithm Fetch below.

Algorithm Fetch(k)

1: Let $N \coloneqq \varnothing$.
2: Sample k parties $p_1, \ldots, p_k \in \mathcal{P}$ weighted w.r.t. α_{p_i} and add these to N.
3: Request data from all N parties and return the union.

The probability that a party does not fetch some data that is already sufficiently spread drops exponentially fast in k. We formalize and prove this in the lemma below.

Lemma 10 (Fetching from Constant Number of Parties)**.** *Let $k \in \mathbb{N}$ and let β be the fraction of weight assigned to parties that are honest and hold some piece of data. The probability that the state returned by $\mathsf{Fetch}(k)$ does not include that data is at most*

$$(1-\beta)^k. \tag{33}$$

Proof. Let D be the set of parties that are honest and hold the data and let X_i for $i \in \{1,\dots,k\}$ be the random variable denoting the ith party that is picked by $\mathsf{Fetch}(k)$. Using the definition of conditional events we have

$$\begin{aligned}\Pr[\text{no picked party is honest and has data}] &= \Pr\left[\bigcap_{i=1}^{k} X_i \notin D\right] \\ &= \prod_{i=1}^{k} \Pr\left[X_i \notin D \;\middle|\; \bigcap_{j<i} X_j \notin D\right].\end{aligned} \tag{34}$$

Furthermore, we have for $i \in \{1,\dots k\}$,

$$\begin{aligned}\Pr\left[X_i \notin D \;\middle|\; \bigcap_{j<i} X_j \notin D\right] &= 1 - \Pr\left[X_i \in D \;\middle|\; \bigcap_{j<i} X_j \notin D\right] \\ &= 1 - \sum_{p_z \in D} \Pr\left[X_i = p_z \;\middle|\; \bigcap_{j<i} X_j \notin D\right] \\ &\leq 1 - \sum_{p_z \in D} \frac{\alpha_z}{\sum_{p_v \in \mathcal{P}} \alpha_v} \\ &= 1-\beta.\end{aligned} \tag{35}$$

Hence, we can conclude that

$$\Pr[\text{no picked party is honest and has data}] \leq (1-\beta)^k. \tag{36}$$

Since $\mathsf{Fetch}(k)$ returns the union of all obtained data, it is sufficient to pick a single honest party that holds the data, which concludes the proof. □

6 Evaluation

To evaluate the practicality of WFF protocol, we implemented a simulator and ran various benchmarks. We now present and analyze the more relevant obtained results.

6.1 Setup

For the evaluation we consider three types of (deterministically determined) weight distributions: constant weight distribution wherein all parties have the same weight; exponential weight distribution wherein the weights of parties form an exponential curve (we believe this type of weight distribution best reflects the distribution of resources in the real world); and few heavy weight distribution wherein a few parties are very heavy, whilst the remainder have very little weight. To be more precise: the constant weight distribution is defined by the number of parties n; the exponential weight distribution is defined by the number of parties n and by the weight ratio r between the heaviest party and the lightest party[9]; the few heavy weight distribution is defined

[9]This means that if one would order parties increasingly by their weights, for $i \in \{1,\dots,n-1\}$, the weight of p_{i+1} is $r^{-(n-1)}$ times the weight of p_i.

by the number of parties n, by the weight ratio r between the heaviest party and the lightest party, and by the number of heavy parties c[10]. Regarding who sends the message: for the exponential weight distribution we consider three possibilities: the heaviest party sends, the lightest party sends, or the party with median weight sends; for the few heavy weight distribution there are only two possibilities: either a heavy party sends the message, or a light party does; for the constant weight distribution, the sender is irrelevant since all parties have the same weight.

We consider three different corruption strategies: random corruption, wherein the adversary picks parties uniformly at random and then corrupts them; light first corruption, wherein the adversary corrupts parties by their weight, starting by the lighter ones; and heavy first corruption, wherein the adversary behaves as for the light first corruption, but instead starts corrupting the heavier parties first. For all corruption strategies, the adversary corrupts as many parties as possible within its corruption budget. This means that even if the adversary cannot corrupt a certain party—because doing so would make the adversary exceed the corruption threshold—it still tries corrupting other parties. When running simulations, for each set of parameters, we make multiple runs. For each run a new corruption set is chosen independently of the corruption sets sampled for prior runs. This means that for the random corruption strategy, the set of corrupted parties is picked fresh in each run. On the other hand, since the light first and heavy first corruption strategies are deterministic, the set of corrupted parties is always the same for each run. Finally, for each run the sender is picked first and cannot be corrupted.

For the constant weight distribution the corruption strategy is irrelevant. For this reason, for this weight distribution we only consider the random corruption strategy.

6.2 Default Configurations

For the simulations, we implemented the WFF protocol. The default number of parties is 1024; the default weight ratio between the lightest party and the heaviest party for both the exponential and few heavy weight distributions is 1.000.000; the default number of heavy parties for the few heavy weight distribution is 10; the default corruption threshold is 50% of the total weight.

For each parameter configuration, we make 200 runs. The average success rate corresponds to the percentage of these runs in which the message is received by all parties. Note, that this definition of a success is differs from the definition of our timely message predicate in two respects:

1. We do not require that message should be delivered before a certain time. The reason is that we have observed that the maximum latency for any configuration which succeeds reasonably often is within a very small factor of δ_{Channel} and hence we consider this practical (see Figures 3b, 4b and 5b).

2. We require that the message is delivered to *all* parties (including dishonest parties). The reason we made this choice is that this makes a greedy corruption strategy optimal (always corrupt as much weight as possible). Otherwise, it could be that leaving a party honest would give a lower success rate than corrupting the party because the protocol then would need to guarantee delivery to this party also. Furthermore, it is any time a run is a success with respect to this definition all honest parties also receives the message. With this choice, we therefore argue that our corruption strategies more likely generalize to an actual worst case corruption strategy, and our success rates therefore corresponds to a lower-bound on the actual success rates.

[10] One can think of the few heavy weight distribution as the constant weight distribution with $n - c$ parties, to which c parties with r times more weight than the original ones are then added.

The maximum latency of configuration is the largest number of hops that a message propagation took in the successful runs (if none of the 200 runs was successful, then we do not plot the latency). The unit of the latency in our plots will therefore be δ_{Channel}.

For the exponential and few heavy distributions, where there can be senders with different weights, we make 200 runs for each case—meaning we make 200 runs for the lightest sender, 200 runs for the median weighted sender, and 200 runs for the heaviest sender, for the exponential weight distribution, and make 200 runs for the lightest sender, and 200 runs for the heaviest sender, for the few heavy weight distribution—and then consider the least average success rate among the different senders. Regarding latency, we simply take the maximum latency among all runs.

6.3 Comparison Against Weight-Oblivious Protocols

Figure 2 shows that our protocol, by making parties pick their targets according to their weight, achieves 100% success rate at a much lower communication cost than the weight-oblivious protocols. In fact, as one can see from the plot, while the our protocol achieves practical security with low message complexity regardless of the weight distribution, security for the weight-oblivious protocol is tied to the weight ratio between the heaviest and the lightest parties.

Note that when the weight ratio between the heaviest and the lightest parties is 1, the exponential weight distribution is the same as the constant weight distribution.

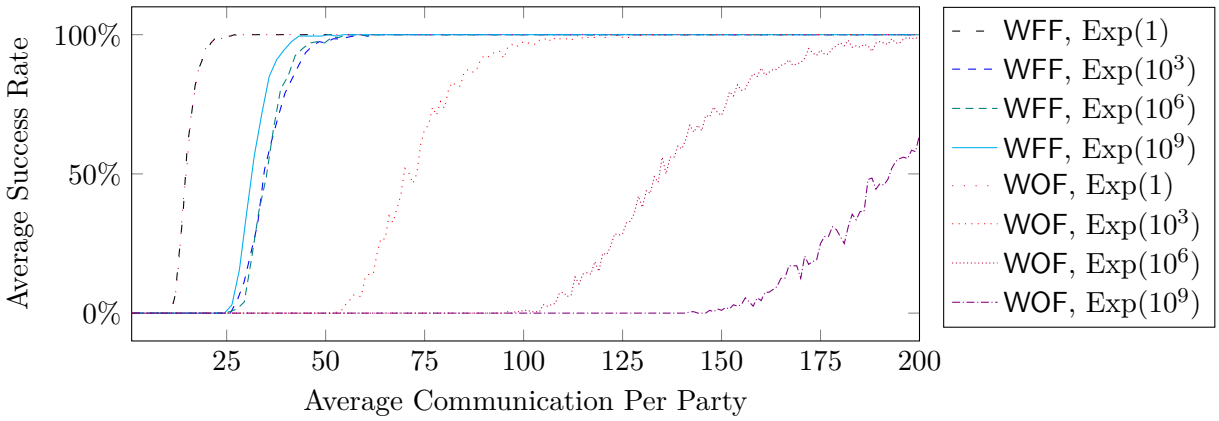
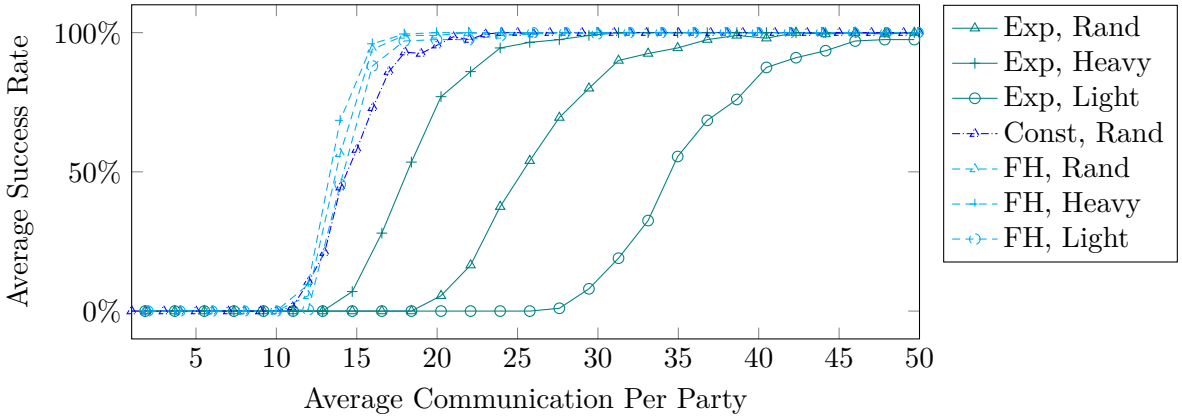


Figure 2: Comparison of WFF protocol against a weight oblivious protocol, for the exponential weight distribution with varying lightest party-heaviest party weight ratios. In the plots, WFF denotes our protocol, and WOF denotes a weight oblivious protocol, namely $\pi_{\text{Flood}}(\mathsf{WFS}(\mathtt{E}, k))$, with the emulation function $\mathtt{E}$ being $\mathtt{E}(p) \coloneqq 1$); Exp denotes the exponential weight distribution, and the parameter is the weight ratio between the lightest and heaviest party. The plot shows how the average success rate varies depending on the average number of messages sent per party, for the two protocols and for the different lightest party-heaviest party weight ratios. We only consider the light first corruption strategy.

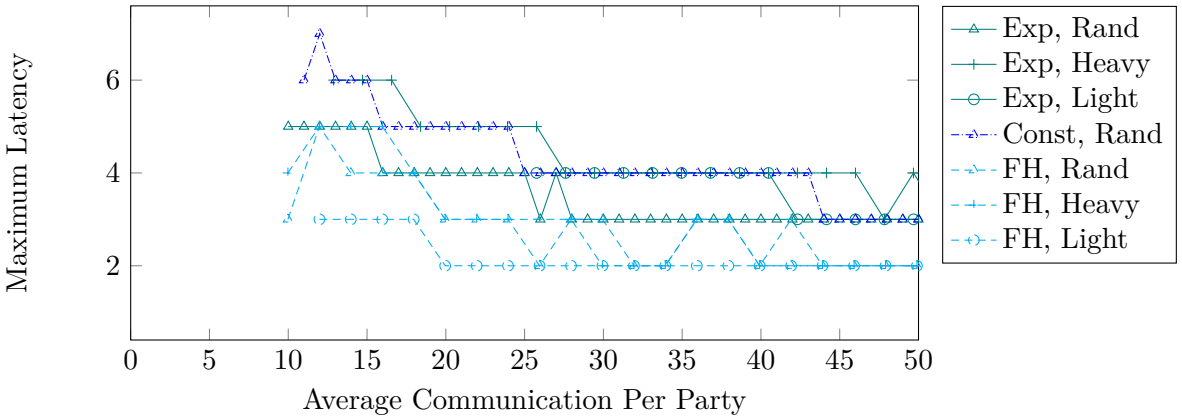
6.4 Performance For Changing Weight Distributions

In Section 3.4.2 we bounded the communication complexity of WFF by $\frac{2 \cdot n \cdot (\log(n)+\kappa)}{\gamma}$ (see Theorem 2) and in Section 4.2 we showed that this is asymptotically optimal for the constant weight distribution for this type of protocol (see Lemma 9). A useful property of our communication complexity bound is that it is independent of the weight distribution. However, as this upper-bound only asymptotically matches the necessary communication complexity for the constant

weight distribution, one could be concerned that are certain weight distribution or corruption strategies that would make WFF perform badly in practice. To show that this is not the case we measured the success rate for sending a single message in WFF(k) and the maximum latency as a function of the communication complexity (induced by adjusting k) for different stake distributions and corruption strategies. The results can be found in Figure 3. We note that the communication complexity for the average success probability to approach 1 is within a small constant factor from each other for all the different weight distributions and corruption strategies. Furthermore, note that for the constant weight distribution WFF(k) simply selects k neighbours uniformly at random and at least $\gamma \cdot n$ of the parties remains honest. Hence, this corresponds to the performance that can be expected by additionally assuming that a certain fraction of the parties remains honest and use flooding protocols tailored to this setting. We emphasize that our protocol only induces marginally larger (within a small constant factor) communication complexity for all the considered weight distributions. Consequently, we conclude that security for our flooding protocol in the weighted setting essentially comes at no cost compared to this more traditional setting.



(a) Average success rate depending on the average communication per party.

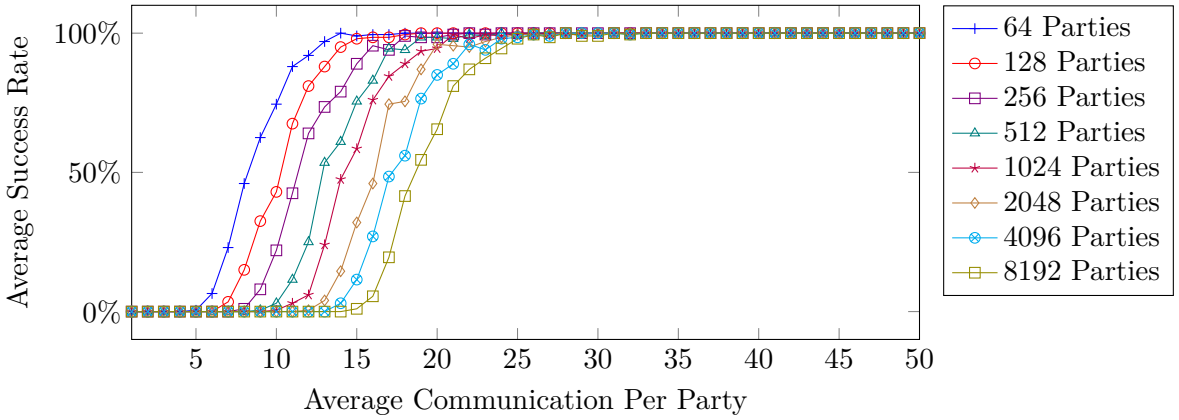


(b) Maximum latency vs average communication per party.

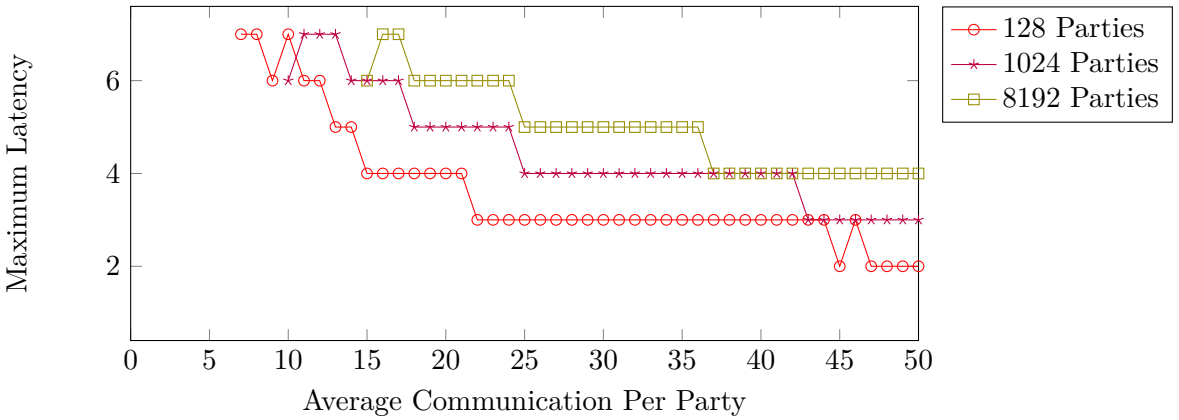
Figure 3: Behavior of WFF protocol for different weight distributions and corruption strategies. We denote the exponential, the few heavy, and the constant weight distributions by Exp, FH, and Const, respectively; and we denote the random, heavy first, and light first corruption strategies by Rand, Heavy, and Light, respectively.

6.5 Scalability

A key feature of message flooding protocols for blockchains is their scalability. To measure the scalability of our proposed protocol, we measured, for different numbers of parties, the average success rate of the protocol depending on the average number of messages each party sends. As Figure 4 shows, our protocol scales very well with the number of parties, not only in terms of the average success rate—which seems to show that as the number of parties double, to achieve a similar success rate parties only have to send one more message, on average—but also in terms of latency, which is very low regardless of the number of parties: for 128 parties, it is only for an average of 22 messages sent per party that the success rate gets to 100%, and for this value the corresponding maximum latency is 4; for 1024 parties, the average number of messages sent per party such that the average success rate is 100% is 27, and the corresponding latency is 4; finally, for 8192 parties, the average number of messages sent per party such that the average success rate is 100% is 32, and the corresponding latency is 5.



(a) Average success rate depending on the number of messages sent per party.



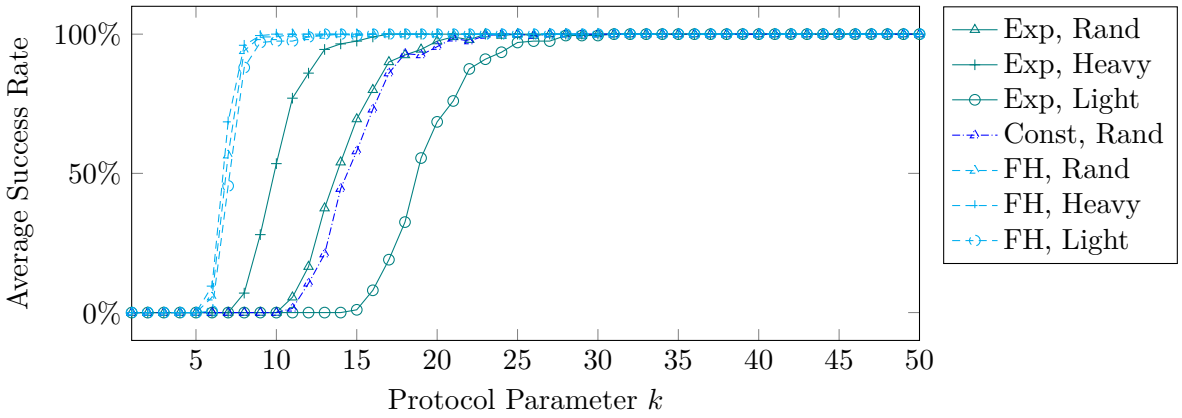
(b) Maximum latency depending on the number of messages sent per party.

Figure 4: Scalability of WFF protocol. For both plots, we only consider the constant weight distribution, a corruption threshold of 50% (meaning, in this case, that the adversary corrupts strictly less than 50% of the parties), and the random corruption.

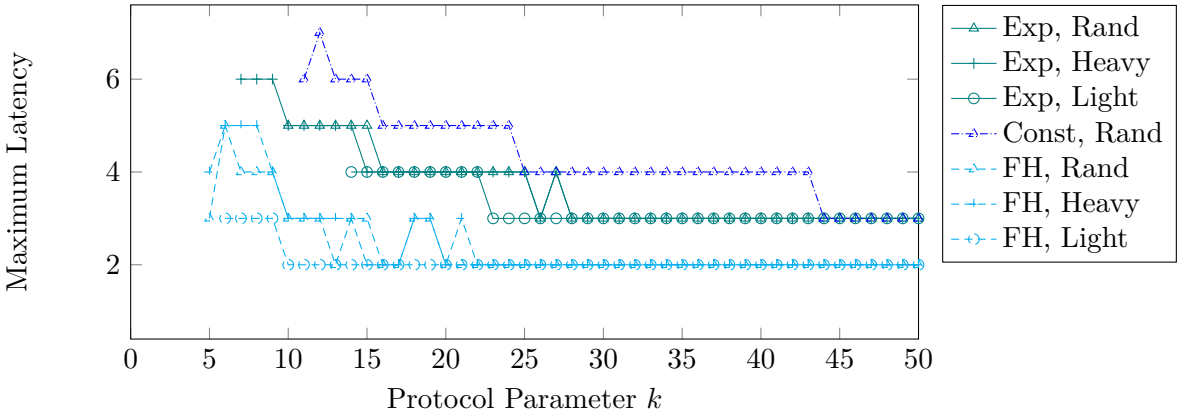
6.6 Estimating Parameter for Practical Security

As already mentioned, the number of messages a party sends is given by its emulation function $\mathtt{E}(p) := \lceil \alpha_p \cdot n \rceil$ multiplied by the protocol parameter k. We now analyze what values one can set k to in order to achieve security in practice. Figure 5 shows, for the different weight distributions and for the different corruption strategies how the average success rate and the maximum latency vary depending on the protocol parameter k.

It is worth mentioning that although, at first sight, our protocols may seem to perform better for the exponential and few heavy weight distributions, this is actually not the case: while it is true that the protocol achieves a higher average success rate and a lower diameter for smaller values of k, for both these weight distributions (but not for the constant weight distribution) the average number of messages sent per party grows by a factor larger than 1 (see Figure 6).



(a) Average success rate depending on the protocol parameter k.



(b) Maximum latency depending on the protocol parameter k.

Figure 5: Behavior of WFF protocol for different weight distributions and corruption strategies. We denote the exponential, the few heavy, and the constant weight distributions by, respectively, Exp, FH, and Const; we denote the random, heavy first, and light first corruption strategies by, respectively, Rand, Heavy, and Light.

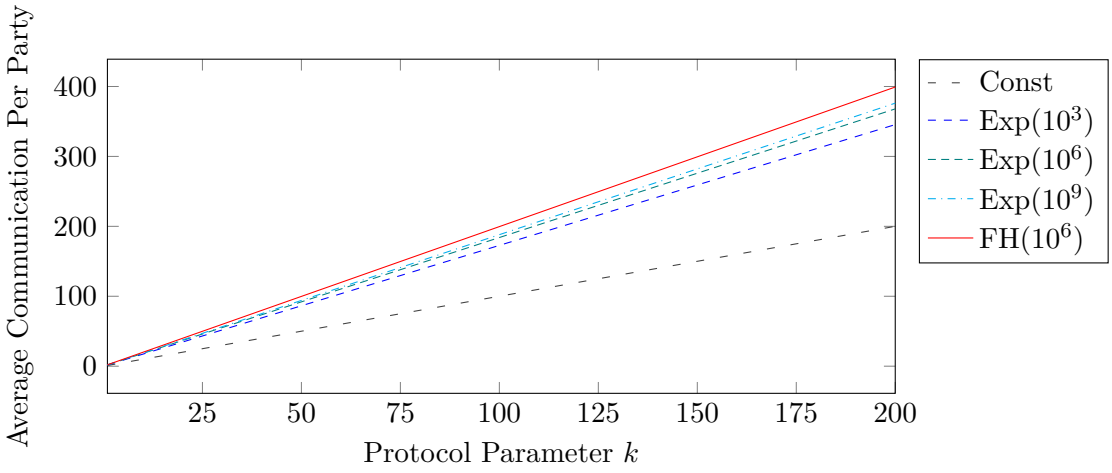


Figure 6: Average number of messages sent per party for WFF protocol, for different weight distributions. In the plot, we denote the exponential, the few heavy, and the constant weight distributions by, respectively, Exp, FH, and Const; the exponential and few heavy weight distributions are parameterized by the weight ratio between the lightest and heaviest party. We only consider 1024 parties, and no corruptions; for the few heavy weight distribution we consider 10 heavy parties. The plot shows how the average number of messages sent per party varies depending on the protocol parameter k, for the different weight distributions.

References

[AMN+20] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. Sync hotstuff: Simple and practical synchronous state machine replication. In *IEEE Symposium on Security and Privacy*, pages 106–118. IEEE, 2020.

[ARV+21] Bithin Alangot, Daniël Reijsbergen, Sarad Venugopalan, Pawel Szalachowski, and Kiat Seng Yeo. Decentralized and lightweight approach to detect eclipse attacks on proof of work blockchains. *IEEE Trans. Netw. Serv. Manag.*, 18(2):1659–1672, 2021.

[AZV17] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *IEEE Symposium on Security and Privacy*, pages 375–392. IEEE Computer Society, 2017.

[BHPS16] Anna Ben-Hamou, Yuval Peres, and Justin Salez. Weighted sampling without replacement, 2016.

[Bol01] Béla Bollobás. *Random Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2 edition, 2001.

[CCG+15] Nishanth Chandran, Wutichai Chongchitmate, Juan A. Garay, Shafi Goldwasser, Rafail Ostrovsky, and Vassilis Zikas. The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In *ITCS*, pages 153–162. ACM, 2015.

[CM19] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777:155–183, 2019.

[DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 66–98. Springer, 2018.

[DMM+20] Thomas Dinsdale-Young, Bernardo Magri, Christian Matt, Jesper Buus Nielsen, and Daniel Tschudi. Afgjort: A partially synchronous finality layer for blockchains. In *SCN*, volume 12238 of *Lecture Notes in Computer Science*, pages 24–44. Springer, 2020.

[DPS19] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *Financial Cryptography*, volume 11598 of *Lecture Notes in Computer Science*, pages 23–41. Springer, 2019.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[FF82] Trevor I. Fenner and Alan M. Frieze. On the connectivity of random m-orientable graphs and digraphs. *Combinatorica*, 2(4):347–359, 1982.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *EUROCRYPT (2)*, volume 9057 of *Lecture Notes in Computer Science*, pages 281–310. Springer, 2015.

[HKZG15] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security Symposium*, pages 129–144. USENIX Association, 2015.

[KMG03] Anne-Marie Kermarrec, Laurent Massoulié, and Ayalvadi J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Trans. Parallel Distributed Syst.*, 14(3):248–258, 2003.

[MHG18] Yuval Marcus, Ethan Heilman, and Sharon Goldberg. Low-resource eclipse attacks on ethereum's peer-to-peer network. *IACR Cryptol. ePrint Arch.*, page 236, 2018.

[MMR99] Dahlia Malkhi, Yishay Mansour, and Michael K. Reiter. On diffusing updates in a byzantine environment. In *SRDS*, pages 134–143. IEEE Computer Society, 1999.

[MNT22] Christian Matt, Jesper Buus Nielsen, and Søren Eller Thomsen. Formalizing delayed adaptive corruptions and the security of flooding networks, 2022.

[MPS01] Dahlia Malkhi, Elan Pavlov, and Yaron Sella. Optimal unconditional information diffusion. In *DISC*, volume 2180 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2001.

[MS03] Yaron Minsky and Fred B. Schneider. Tolerating malicious gossip. *Distributed Comput.*, 16(1):49–68, 2003.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[NKMS16] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *EuroS&P*, pages 305–320. IEEE, 2016.

[PS17a] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *PODC*, pages 315–324. ACM, 2017.

[PS17b] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, volume 91 of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[PS18] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2018.

[RT19] Elias Rohrer and Florian Tschorsch. Kadcast: A structured approach to broadcast in blockchain networks. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019, Zurich, Switzerland, October 21-23, 2019*, pages 199–213. ACM, 2019.

[TCM+20] Muoi Tran, Inho Choi, Gi Jun Moon, Anh V. Vu, and Min Suk Kang. A stealthier partitioning attack against bitcoin peer-to-peer network. In *IEEE Symposium on Security and Privacy*, pages 894–909. IEEE, 2020.

[TLP20] Georgios Tsimos, Julian Loss, and Charalampos Papamanthou. Gossiping for communication-efficient broadcast. Cryptology ePrint Archive, Report 2020/894, 2020. https://ia.cr/2020/894.

[XGS+20] Guangquan Xu, Bingjiang Guo, Chunhua Su, Xi Zheng, Kaitai Liang, Duncan S. Wong, and Hao Wang. Am I eclipsed? A smart detector of eclipse attacks for ethereum. *Comput. Secur.*, 88, 2020.

[YM13] Osman Yagan and Armand M. Makowski. On the scalability of the random pairwise key predistribution scheme: Gradual deployment and key ring sizes. *Perform. Evaluation*, 70(7-8):493–512, 2013.

[ZL19] Shijie Zhang and Jong-Hyouk Lee. Eclipse-based stake-bleeding attacks in pos blockchain systems. In *BSCI*, pages 67–72. ACM, 2019.

[ZTA21] Haofan Zheng, Tuan Tran, and Owen Arden. Total eclipse of the enclave: Detecting eclipse attacks from inside tees. In *IEEE ICBC*, pages 1–5. IEEE, 2021.

A Basic Equalities and Inequalities

Lemma 11 (Binomial formula)**.** *For $x, y \in \mathbb{R}$ and $n \in \mathbb{N}$*

$$(x+y)^n = \sum_{k=0}^{n} x^{n-k} \cdot y^k.$$

Lemma 12 (Bernoulli's inequality)**.** *For $r \in \mathbb{R} \setminus (0,1)$ and $x \geq -1$*

$$1 + r \cdot x \leq (1+x)^r .$$

Lemma 13 (Exponential inequality). *For $y \geq 1$ and $|x| \leq 1$*

$$\left(1 + \frac{x}{y}\right)^y \leq e^x.$$

Lemma 14 (Chernoff bound). *Let $X_1, \ldots, X_n$ be independent random variables with $X_i \in \{0, 1\}$ for all i, and let $\mu \coloneqq E[\sum_{i=1}^n X_i]$. We then have for all $\delta \in [0, 1]$,*

$$\Pr\left[\sum_{i=1}^n X_i \leq (1-\delta)\mu\right] \leq e^{-\frac{\delta^2\mu}{2}} \quad \textit{and} \quad \Pr\left[\sum_{i=1}^n X_i \geq (1+\delta)\mu\right] \leq e^{-\frac{\delta^2\mu}{3}}.$$

B Erdős–Rényi Graph Results

For completeness we restate a bound on the diameter of Erdős–Rényi graphs from [MNT22].

Lemma 15 (Erdős–Rényi graphs with logarithmic diameter [MNT22]). *Let $\eta \in \mathbb{N}$, $d \in \mathbb{R}$, $\mu, \delta_1, \delta_2 \in [0, 1]$, and $\rho \coloneqq \frac{d}{\eta}$. Furthermore, let $\nu \in \mathbb{R}$, let $G \xleftarrow{\$} \mathcal{G}_{\mathrm{ER}}(\eta, \rho)$, and let $t_0 \coloneqq \frac{\log\left(\frac{\mu\eta}{(1-\delta_1)d}\right)}{\log((1-\delta_2)\nu)} + 1$. If*

$$e^{-d\mu} + \frac{\mu\nu}{1-\mu} \leq 1 \quad \textit{and } (1-\delta_2)\cdot\nu > 1, \tag{37}$$

then

$$\Pr[\neg\phi_{Diam}(G, t_0+1)] \leq \eta \cdot \left(e^{-\frac{\delta_1^2 d}{2}} + t_0 e^{-\frac{\delta_2^2\nu(1-\delta_1)d}{2}}\right) + e^{-\eta\cdot(d\mu^2-2)}. \tag{38}$$

Following the instantiation of variables from [MNT22] this leads to the following corollary.

Corollary 5. *Let $\eta \in \mathbb{N}$, $d \in [7, \infty]$, let $\rho \coloneqq \frac{d}{\eta}$, and let $G \xleftarrow{\$} \mathcal{G}_{\mathrm{ER}}(\eta, \rho)$. Then*

$$\Pr\left[\neg\phi_{Diam}\left(G, 7\cdot\log\left(\frac{\eta}{2\cdot d}\right)+2\right)\right] \leq \eta\cdot\left(e^{-\frac{d}{18}} + \left(6\cdot\log\left(\frac{\eta}{2\cdot d}\right)+1\right)\cdot e^{-\frac{7\cdot d}{108}}\right) + e^{-\eta\cdot\left(\frac{d}{9}-2\right)}. \tag{39}$$

Proof. We set

$$\delta_1 \coloneqq \delta_2 \coloneqq \mu \coloneqq \frac{1}{3} \quad \text{and} \quad \nu \coloneqq \frac{7}{4}.$$

The bound now follows from Lemma 15 as the precondition is fulfilled. □