Asymptotically Optimal Message Dissemination with Applications to Blockchains

Chen-Da Liu-Zhang[1], Christian Matt[2], and Søren Eller Thomsen[3]

[1]NTT Research, Sunnyvale, USA
chen-da.liuzhang@ntt-research.com
[2]Concordium, Zurich, Switzerland
cm@concordium.com
[3]Concordium Blockchain Research Center, Aarhus University, Denmark
sethomsen@cs.au.dk

December 14, 2022

Abstract

Messages in large-scale networks such as blockchain systems are typically disseminated using flooding protocols, in which parties send the message to a random set of peers until it reaches all parties. Optimizing the communication complexity of such protocols and, in particular, the per-party communication complexity is of primary interest since nodes in a network are often subject to bandwidth constraints. Previous flooding protocols incur a communication complexity of $\Omega(l \cdot n \cdot (\log(n) + \kappa))$ bits to disseminate an l-bit message among n parties with security parameter κ. In this work, we present the first flooding protocols with optimal total communication complexity of $O(l \cdot n)$ bits and per-party communication of $O(l)$ bits. We further show how our protocols can be instantiated provably securely in proof-of-stake blockchains.

To demonstrate that one of our new protocols is not only asymptotically optimal but also practical, we perform several probabilistic simulations to estimate the concrete complexity for given parameters. Our simulations show that our protocol significantly improves the per-party communication complexity over the state-of-the-art for practical parameters. Hence, for given bandwidth constraints, our results allow to, e.g., increase the block size, improving the overall throughput of the blockchain.

Contents

1 Introduction

Current blockchain protocols rely on the availability of a multicast network that allows any party to communicate with all other parties in the network, and therefore the security and efficiency of the blockchain protocol are heavily influenced by its underlying multicast network.

In typical blockchain protocols, including Bitcoin [Nak08] and Ethereum [W+14], such multicast networks are efficiently implemented via a flooding protocol [KMG03, MNT22, LMM+22], which lets the sender select a set of neighbors randomly and forward the message to these parties, who will forward the messages to another randomly chosen set of neighbors and so on. It was shown in [KMG03] that one needs an expected neighborhood size of $\Omega(\log(n) + \kappa)$, where n is the number of parties and κ is a security parameter, for the message to reach all parties with overwhelming probability in κ. As a consequence, current flooding protocols incur $\Omega(l \cdot (\log(n) + \kappa))$ bits of per-party communication (in total $\Omega(l \cdot n \cdot (\log(n) + \kappa))$ bits), for an l-bit message. A trivial lower bound on the total communication complexity is $\Omega(l \cdot n)$ since all n parties need to receive the message. This, in turn, implies that (the maximal) per-party communication must be $\Omega(l)$ bits. This leaves a gap between the lower bounds and what current flooding protocols achieve. For practical blockchain systems where messages contain large blocks (e.g., around 1MB), the incurred communication constitutes one of the main bottlenecks. We therefore ask the following question:

> *Is there a flooding protocol that incurs the optimal total communication of $O(l \cdot n)$ bits, where each party communicates $O(l)$ bits?*

We answer this question in the affirmative by providing two highly robust flooding protocols for n parties with a success rate overwhelming in the security parameter κ. Our protocols require no setup and are practically efficient even for small number of parties and message length. Moreover, we show how to extend our protocols to the weighted-setting, where each party is assigned a positive weight of a certain resource (such as stake), and the adversary can corrupt any set of parties accumulating a constant fraction of the total resource. More details follow below.

1.1 Contributions

Warm up: Optimal flooding with a linear neighborhood and constant diameter. We first present a simple protocol ECCast[1], that requires each party to send messages to all other parties, but it achieves a constant diameter of just 2. The protocol works by letting the sender of a message divide their message into n (the number of parties) different shares using an erasure-correcting code, and then send a unique share to each party. When a party receives such a share, they will forward it to *all* other parties. Once a party receives sufficiently many shares, they will be able to reconstruct the original message.

Theorem 1 (ECCast (informal))**.** *For n parties, ECCast ensures asymptotically optimal flooding with a diameter of 2 and an overwhelming success probability in κ, for message of length at least $\Omega(n \cdot (\log(n) + \kappa))$ and at least a constant fraction of the parties remaining honest.*

Even though this protocol requires each party to send messages to all other parties, we believe that it has wide applications as it allows one to "balance" the incurred communication among parties, at the cost of doubling the diameter (over the naive protocol in which the sender directly sends the whole message to all parties). In fact, independently and concurrently with our work, Kaklamanis, Yang and Alizadeh [KYA22] use such techniques to speed up the Hotstuff consensus protocol [YMR+19].

[1] ECCast from the use of Erasure-Correcting codes and each party multicasting messages to all parties.

Table 1: Comparison of flooding for messages of length l among n parties where a constant fraction of parties is honest.

Property	Naive	[MNT22, LMM+22]	ECFlood (this work)	ECCast (this work)
Min. message length	1	1	$\Omega((\log n + \kappa)(\log\log n + \kappa))$	$\Omega(n \cdot (\log(n) + \kappa))$
Max. neighbors	$n-1$	$O(\log(n) + \kappa)$	$O(\log(n) + \kappa)$	$n-1$
Max. per-party comm.	$l \cdot (n-1)$	$O(l \cdot (\log(n) + \kappa))$	$O(l)$	$O(l)$
Total comm.	$l \cdot (n-1)$	$O(l \cdot n \cdot (\log(n) + \kappa))$	$O(l \cdot n)$	$O(l \cdot n)$
Diameter	1	$O(\log(n))$	$O(\log(n))$	2

Optimal flooding with a logarithmic neighborhood and diameter. We then present the protocol ECFlood,[2] which requires each party to connect to only $O(\log(n) + \kappa)$ other parties and use only $O(l)$ of per-party communication.

At a high level, the protocol works by letting the sender of a message divide their message into a number of shares μ. Each of these shares will then be sent to each party with an independent probability ρ. When a party receives such a share, they will then again forward it to all other parties with the same independent probability ρ. Once a party receives sufficiently many shares, they will be able to reconstruct the original message.

Theorem 2 (ECFlood (informal))**.** *For n parties and a security parameter κ, there are $\mu = O(\log(n) + \kappa)$ and $\rho = O(n^{-1})$ such that ECFlood ensures asymptotically optimal flooding with a logarithmic diameter and an overwhelming success probability in κ, for messages of length at least $\Omega((\log(n) + \kappa) \cdot (\log(\log(n)) + \kappa))$ and at least a constant fraction of the parties remaining honest.*

In particular, it is worth noting that ECFlood shaves a factor of $\log(n) + \kappa$ off both the communication complexity and the per-party communication over previous best-known constructions. This is done while keeping both the size of the neighborhood and the diameter at the same level as these previously best-known constructions. We further note that ECFlood requires no trusted setup but merely relies on a weak cryptographic accumulation scheme, which can be realized efficiently from standard cryptographic assumptions.

We summarize the properties of ECFlood and ECCast and compare them to other flooding protocols with similar robustness in Table 1, where the "naive" protocol refers to the protocol where the sender simply sends the message to all other parties. Note that both protocols are secure for any message size (and achieve optimal communication for sufficiently long messages).

Probabilistic simulations for ECFlood. While the theoretical analysis of ECFlood shows that our protocol is asymptotically optimal, we use probabilistic simulations to evaluate its practical efficiency. The main results of our simulations are shown in Figures 1 and 2. The parameter d corresponds to the expected number of parties each share is sent to by every party, i.e., $\rho = d/n$. Figure 1 shows that increasing d also increases the redundancy (i.e., the per-party communication complexity divided by the message length l) and thus the communication complexity. On the other hand, Figure 2 shows that increasing d decreases the latency. Since there are μ shares to be sent to d parties, the total number of neighbors per parties is $\mu \cdot d$. Figure 1 further shows that increasing the number of shares μ decreases the redundancy.

With these tradeoffs in mind, we can compare our results to the state-of-the-art provably secure flooding protocol [LMM+22]. The simulations in [LMM+22, Figure 3] show that in order to succeed 99% of the time for 8192 parties, one has to, in their protocol, let each party forward

[2] ECFlood from the use of Erasure-Correcting codes in the flooding protocol.

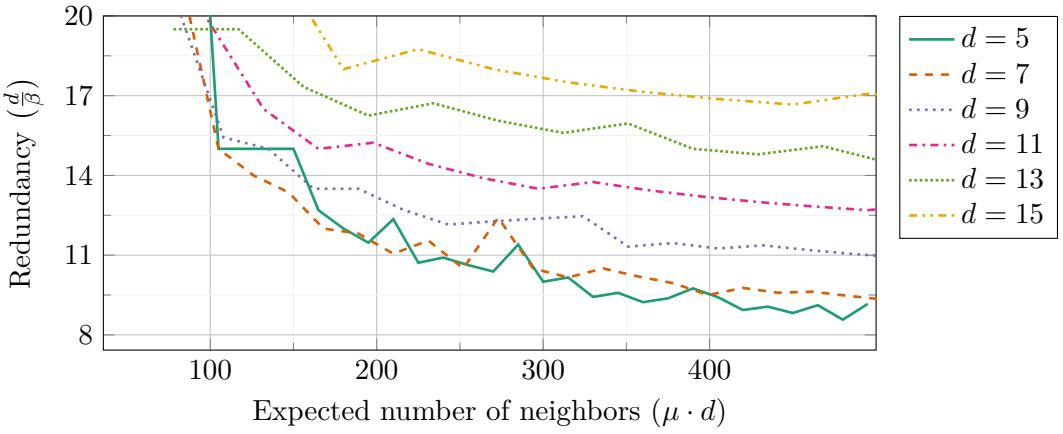

Figure 1: Results for simulations of $\mathsf{ECFlood}(\frac{d}{n})$ for different values of d, a fixed number of parties $n = 8192$, and a variable number of shares μ. The graphs show the redundancy of the protocol as a function of the expected number neighbors. The number of shares is incremented in steps of 3.

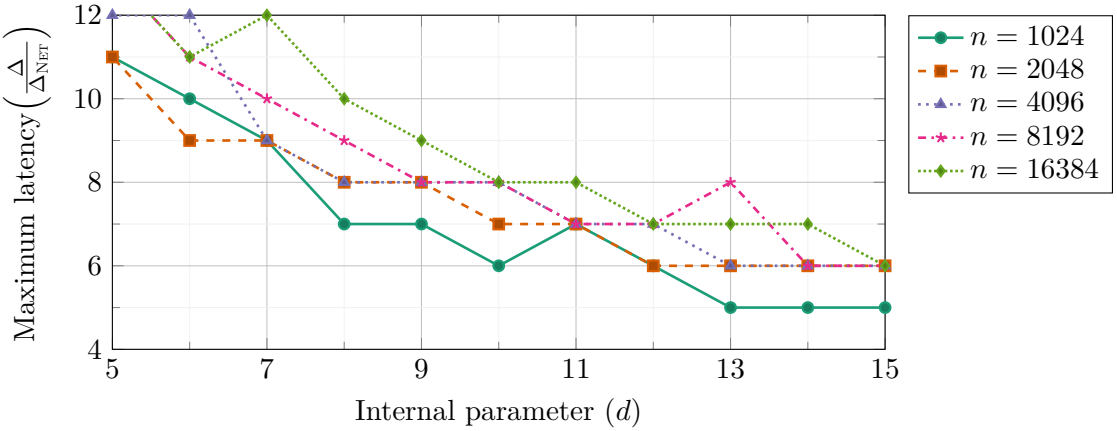

Figure 2: Results for simulations of $\mathsf{ECFlood}(\frac{d}{n})$ for different values of d and n, but with a fixed number of shares $\mu = 30$. The graphs show the maximal latency before any party has received enough shares to be able to reconstruct for different values of d.

the entire message to around 28 neighbors, resulting in a latency of 6 network delays with redundancy 28. $\mathsf{ECFlood}$ for $d = 15$ achieves the same latency with a redundancy below 20 for 200 neighbors per party. If a slightly larger diameter is acceptable, $\mathsf{ECFlood}$ with $d = 9$ has a worst-case latency of 8 network delays and a redundancy of less than 14 for 200 neighbors per party. This means that even though parties have to connect to more neighbors in $\mathsf{ECFlood}$, the total amount of data sent is less than half compared to the state of the art for these parameters. We note that the number of connections is also a limiting factor, but 200 is still practical; e.g., Bitcoin clients by default have up to 125 connections [GRKC15].

Flooding in the weighted setting. We leverage the idea of emulation from [LMM+22] to make a general transformation from a flooding protocol that is secure, assuming a constant fraction of the parties behaves honestly, to a flooding protocol that is secure, assuming that a constant fraction of publicly assigned weights behaves honestly. We do so by introducing the protocol $\mathsf{Flood2WeightedFlood}$ that reduces the task of finding a secure flooding protocol among

an actual set of parties to a secure flooding protocol for an emulated set of parties. In more detail, let α_p denote the fraction of total weight assigned to a party p, let there be n parties in total, and let the total fraction of weight given to honest parties be given by $\widetilde{\gamma}$. We observe that by letting each party p emulate $\lceil \alpha_p \cdot n \rceil$ parties, the fraction of these emulated parties that will behave honestly is lower bounded by $\widetilde{\gamma} \cdot 2^{-1}$. Hence, if we assume a constant fraction of honest weight, there will also be a constant fraction of honest emulated parties. This implies that we can translate our protocols ECFlood and ECCast (as they will work for any constant fraction of parties), using Flood2WeightedFlood, also to work assuming only a constant fraction of honest weight. This happens while only increasing the total communication complexity by a factor of at most 4. The per-party communication will, however, in this case, be proportional to the amount of weight each party is assigned. We note that in [LMM+22, Corollary 4], it was shown that it is inherent for the weighted setting that parties with a large fraction of weight must send more messages.

1.2 Technical Overview

As discussed above, prior works that let each party forward a message to a random subset of neighbors (see, e.g., [KMG03, MNT22, LMM+22]) need each party to connect to $\Omega(\log(n) + \kappa)$ parties to ensure that the message is propagated to all parties with overwhelming probability in the security parameter κ. Intuitively, the term κ is needed to make the probability that an individual party has no honest neighbors negligible. Further, to ensure that the probability that no party is unlucky is negligible, the additional $\log(n)$ neighbors are needed. It is therefore not possible to decrease the number of needed connections of such flooding protocols.

In order to further improve the communication complexity, we focus instead on reducing the number of bits sent to each of these neighbors. For that, we deviate from the above approach and design our flooding protocol in two steps. First, we consider a *weak* flooding protocol that ensures that with a constant probability, a constant fraction of the parties receives the message. Secondly, we introduce a compiler that lifts a weak flooding protocol to a complete flooding protocol that guarantees delivery to all parties with overwhelming probability.

Flooding amplification. The protocol compiler WeakFlood2Flood splits a message into a number of shares μ using erasure-correcting codes and makes use of a weak flooding protocol to distribute each of these shares. Since the shares are created using erasure-correcting codes, it is not necessary for each party to receive all shares to reconstruct the original message.

An apparent attack on such protocol would be for an adversary to try to inject "fake" shares into the set of shares honest parties try to reconstruct the message from. We prevent this by using a cryptographic accumulation scheme, to prove that a particular share is part of the original shares. Such accumulator can be implemented efficiently, e.g., using Merkle trees or signature schemes.

More concretely, let the reconstruction threshold be $\tau = \xi \cdot \mu$, for some constant ξ. Using standard erasure-correcting codes (e.g., Reed-Solomon codes), this can be obtained with a share size of $O(l \cdot \tau^{-1})$. In order to achieve a flooding protocol with optimal communication, we need to ensure that 1) each party receives τ shares and 2) each instance of weak flooding only incurs constant overhead with respect to the size of each share (i.e., $O(l \cdot \tau^{-1})$ bits) of per-party communication.

Using the Chernoff bound, one can show that if there is a constant independent probability for each party to receive each sent share, then all parties receive a constant fraction of the shares with overwhelming probability when at least $\log(n) + \kappa$ shares are sent.

Hence, the task of finding an asymptotically optimal flooding protocol is reduced to finding

a weak flooding protocol that ensures delivery with a constant independent probability for each input message and sends each input message to only a constant number of parties.

A weak flooding protocol. Our candidate for a weak flooding protocol is the protocol $\mathsf{ERFlood}(\rho)$ that lets each party forward each message to each party with an independent probability ρ.

Previous works [KMG03] showed that the probability that there is an isolated party for $n \to \infty$ when $\rho = \frac{\log(n)+c+o(1)}{n}$ for some constant c is given by $1 - e^{-e^c}$. This means that one needs to set $\rho = \Omega(\log(n) \cdot n^{-1})$ to have a constant success probability for all parties to receive the message. As a consequence, the expected size of each neighborhood would be $\Omega(\log(n))$, which is too much communication.

To overcome this, we observe that we only need that the probability that any fixed party receives the message is constant. Using a novel analysis of the protocol, we prove that by sending the message to only a constant number of neighbors, there is a constant probability that the message reaches a constant fraction of all parties.

1.3 Model

Our results are proven for a static set of parties connected by unauthenticated point-to-point channels that immediately leak any message sent to the adversary. Our analysis additionally uses an upper bound on the time it takes to send a message through each channel, but this is not required to be known before our protocols are deployed.

On composable security. We prove that our protocols implement a property-based definition of flooding. It is, therefore, not automatically guaranteed that our results compose with other constructions as it would have been if they were proven within a composable security framework such as UC [Can20]. However, note that none of our protocols has any secrecy, i.e., all inputs given to any honest party are immediately leaked to the adversary. As shown in [MNT22], it is therefore straightforward to simulate such protocols because the simulator can run the original flooding protocol with full knowledge of all inputs. Based upon this, we believe that it is straightforward to show that our flooding protocols UC-realize a flooding functionality as the one provided in [MNT22].

On security against adaptive adversaries. Our results are proven for all adversaries that can statically corrupt all but a constant fraction of the parties (except for the protocol $\mathsf{Flood2WeightedFlood}$, where we naturally require that a constant fraction of the weight remains honest). However, even stronger results hold. The protocol ECCast is secure against an adaptive adversary if we assume that the adversary cannot retract messages that are already sent (often referred to as the *atomic message send model*). Unfortunately, as observed by Matt et al. [MNT22], this is not sufficient for any protocol with a sublinear neighborhood because an adversary can then simply corrupt the entire neighborhood of the sender and thereby prevent delivery of the message. This rules out that $\mathsf{ECFlood}$ can be proven secure using only this assumption.

To circumvent this apparent impossibility of security against adaptive adversaries for flooding protocols, Matt et al. [MNT22] introduced the model of δ-delayed adaptive adversaries, where it takes a certain time from when an adversary decides to corrupt a party until the adversary gains control of this party. All of our proofs for $\mathsf{ECFlood}$ go through if it is assumed that an adversary is delayed for the entire delivery time of the protocol, as then the set of parties controlled by

an adversary will be independent of the neighborhoods chosen for the delivery of a particular message. The setting will, thereby, essentially be static for each specific message.

We note that it may not be practically feasible to require each peer to resample their neighbors for each message, as suggested by our protocol ECFlood. If one is willing to assume an adversary delayed for an extended period of time, then our protocol can be run securely without resampling neighbors for each message but instead only resampling neighbors at specific time intervals of length at least the corruption delay of the adversary.

On the weighted model. Our protocol Flood2WeightedFlood compiles any protocol secure assuming a constant fraction of parties to one that is secure assuming only a constant fraction of publicly assigned weights remains under honest control. This assumption can be realized for *Proof-of-Stake* based protocols [DGKR18, DPS19, CM19] where the protocol relies on a constant fraction of the stake behaving honestly. As the stake is publicly available in the maintained ledger of such protocols, it is immediate that it can be used directly as public weights.

As noted by [LMM+22], committee selection techniques [PS17b, PS18] can be used to instantiate weights from other resource assumptions such as computational power.

1.4 Related Work

Flooding protocols. Flooding protocols are used to implement so-called multicast networks, which allow a party to distribute a message among a set of parties within some prescribed time. Current flooding protocols (as in Bitcoin [Nak08], Ethereum [W+14], etc.) are typically implemented via a forwarding mechanism, where in order for a party to distribute a message, the party simply selects a random subset of neighbors, who then forward the message to their neighbors and so on.

The security of such a protocol relies on the fact that the graph induced by the neighbor selection procedure among honest parties is connected. Kermarrec, Massoulié and Ganesh [KMG03] showed that when choosing each neighbor with probability ρ in a setting with up to $t = (1-\gamma) \cdot n$ corruptions (out of n parties), it is necessary that $\rho > \frac{\log(n)+\kappa}{\gamma \cdot n}$ to ensure that messages are delivered to all honest parties with overwhelming probability in κ.

Matt, Nielsen, and Thomsen [MNT22] formally proved security of such a flooding protocol against a so-called delayed adaptive adversary (where it takes a certain delay for the adversary to gain control over a party) corrupting any fraction of the total number of parties. In a followup work [LMM+22], Liu-Zhang, Matt, Maurer, Rito, and Thomsen gave the first protocol that remains secure 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. We adapt the techniques from Liu-Zhang, Matt, Maurer, Rito, and Thomsen and provide a general procedure for obtaining a flooding protocol for the weighted setting from one secure in the none weighted setting. In particular, this allows our protocols to be used in the weighted setting.

The protocols of [KMG03, MNT22, LMM+22] incur a total communication of $O(l \cdot n \cdot (\log(n) + \kappa))$ bits, for a message of size l. In contrast, our protocols incur the (asymptotically) optimal total communication of $O(l \cdot n)$.

Coretti, Kiayias, Moore, and Russell [CKMR22] considered the problem of designing a message diffusion mechanism based on the majority of honest stake assumption tailored specifically for the Ouroboros Praos consensus protocol [DGKR18]. However, their flooding protocol achieves a weaker guarantee in that it allows a certain set of honest parties to be eclipsed. In contrast, our work focuses on flooding protocols that guarantee delivery to all honest parties.

Another line of work seeks to improve on the efficiency of flooding protocols for blockchains by applying structured approaches and heuristics [FOA16, RT19, VT19]. However, the behavior

of these protocols under byzantine corruptions is not documented, and our focus is on provably secure protocols. We do, therefore, not comment on this line of work further.

Agreement primitives for long messages. A significant line of work is dedicated to building broadcast and Byzantine agreement primitives for long messages for different thresholds, setups, and assumptions, starting from the work of Turpin and Coan [TC84]. Many subsequent works achieve communication complexity $O(l \cdot n + \text{poly}(n, \kappa))$ (see, e.g., [NRS+20, FH06, GP16, BLZLN22]). We note that techniques similar to those we use for our ECCast protocol were used within agreement protocols in [NRS+20].

In all these works, parties communicate to all other parties (so the neighborhood size is $n-1$). In contrast, we provide ECFlood where each party communicates to only $O(\log(n)+\kappa)$ neighbors.

2 Model and Preliminaries

In this section, we define the model, in which we prove our results, specify the primitives our constructions rely on and give suggestions for how to instantiate these primitives. Additionally, we define notation and basic bounds that we will use for our proofs.

2.1 Parties, Adversary and Communication Network

We consider a set of n parties $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$. For simplicity, we assume an adversary that can statically corrupt a set of parties such that only a subset $\mathcal{H} \subseteq \mathcal{P}$ will behave honestly.[3] We will use h to denote a bound on the size of $\mathcal{H}$. For the remainder of the paper, we will assume that $|\mathcal{H}| \geq h$. We will use $\gamma := \frac{h}{n}$ to denote the fraction of parties guaranteed to be honest, and assume this to be a constant.

We assume that all parties are connected pairwise through unauthenticated point-to-point channels and let Δ_{NET} denote an upper bound on the delivery time for the underlying point-to-point channels.

2.2 Primitives

Erasure correcting codes. In our protocols we make use of a special type of weak error-correcting code that is only able to tolerate a certain number of erasures. We refer to these as erasure-correcting codes.

Definition 1 (Erasure Correcting Code Scheme)**.** Let $\mu \in \mathbb{N}$ be the number of shares, and let $e \in \mathbb{N}$ be the number of erasures that are to be tolerated. A pair of algorithms ζ is a (μ, e)-erasure-correcting-code-scheme (abbreviated (μ, e)-ECCS) if it consists of two algorithms:

- ζ.Enc: An encoding algorithm that takes a message $m \in \{0,1\}^*$ and produces a sequence of shares $s_1, \ldots, s_\mu$.
- ζ.Dec: A decoding algorithm that if a sequence of shares $s'_1, \ldots, s'_\mu$ s.t. it holds for at least $\mu - e$ of them that $s'_i = s_i$ and for the remaining $s'_i = \perp$ is input, then the original message is m is returned.

We will use the notation ζ.`ShareSize`(l) for a function that bounds the size of each share when a message of length l is encoded.

[3] One can extend our protocols to handle so-called delayed adaptive adversaries, using techniques presented in [MNT22].

Standard Reed-Solomon codes [RS60] can be used to instantiate a (μ, e)-ECCS in a straightforward manner. That is, for a message m with length l (where $l \geq \mu - e$), let each share be an element in a Galois Field of order 2^a for $a = \lceil \max(\log(\mu - 1), \log(\frac{l}{\mu - e})) \rceil$. If ζ is such a scheme, we will have for a message where $a = \lceil \log(\frac{l}{\mu - e}) \rceil$ that

$$\zeta.\texttt{ShareSize}(l) = O\Big(\frac{l}{\mu - e}\Big). \tag{1}$$

This implies that the total bitlength of the shares will be $O(l \cdot \frac{\mu}{\mu - e})$. We note that it has been shown that the encoding and decoding of such codes be done in $O(2^a \cdot a^2)$ time [Did09].

Weak cryptographic accumulators. We will in the paper make use of a *weak* version of a *static positive* accumulator. *Weak* refers to that we only require collision-freeness and correctness to hold for honestly generated accumulators, *static* refers to that we do not need the set of accumulated values to be dynamically extendable, and *positive* means that we only need to prove membership of an accumulator (in particular we do not need to prove that an element is not a part of the accumulator). We define this below.

Definition 2 (Weak Static Cryptographic Accumulation Scheme)**.** A pair of algorithms α is a *weak static cryptographic accumulation scheme* (abbreviated WSCAS) if it consists of two algorithms:

- $\alpha.\mathsf{Accumulate}(\{m_1, \ldots, m_\eta\})$: An algorithm for accumulating a set of values $\{m_1, \ldots, m_\eta\}$. It returns an accumulated value z and a sequence of proofs $\pi_1, \ldots, \pi_\eta$ where π_i can be used to prove that m_i is in the accumulated value z where each $m_i \in \{0,1\}^*$.
- $\alpha.\mathsf{Verify}(m, \pi, z)$: A function that checks if a proof π proves that a message m was in the set of elements used to create the accumulated value z.

With the following properties:

Completeness: All honestly generated proofs are accepted by $\alpha.\mathsf{Verify}$.

Collision-freeness: No polynomial-time adversary can find a set of values $M \coloneqq \{m_1, \ldots, m_\eta\}$, a value $m' \notin M$, and a proof π such that $\alpha.\mathsf{Verify}(m', \pi, z) = \top$ for $z \leftarrow \alpha.\mathsf{Accumulate}(M)$.

See [BP97] for the original formal definition of collision-freeness and [ÖMBS21] for an overview of accumulator constructions.

We use the notation $\alpha.\texttt{AccSize}$ for a bound on the size of the accumulated value and $\alpha.\texttt{ProofSize}(\eta)$ for a function that bounds the size of each proof as a function of the number of messages accumulated η.

Because we only require collision-freeness for honestly generated accumulators, a WSCAS scheme can be efficiently instantiated using a regular signature scheme by letting the accumulated value z be the public verification key, and a proof for a message be a signature of that message. For suitable signature schemes this yields

$$\alpha.\texttt{AccSize} = O(\kappa) \text{ and } \alpha.\texttt{ProofSize}(\eta) = O(\kappa). \tag{2}$$

The same complexity can be achieved by basing α on RSA accumulators [BdM93] or bilinear accumulators [Ngu05]. To avoid generating keys or a setup assumption, one can also use Merkle Trees [Mer89] as accumulators, at the cost of slightly increasing the proof size to $\alpha.\texttt{ProofSize}(\eta) = O(\log(\eta) \cdot \kappa)$.

Flooding. A flooding protocol allows a set of parties to send messages to each other subject to certain delivery guarantees. Our definition is based on the one presented in [LMM+22] with only minor differences. Informally, that is that we want that each message input delivered to all parties with a probability that is overwhelming in the security parameter.

Definition 3 (Flooding)**.** 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 *strong* Δ-flooding protocol if when a message m is input to an *honest* party at time t, then by time $t + \Delta$ there is a probability overwhelming in the security parameter κ that all other honest parties output m.

In contrast to previous definitions [LMM+22], we do not require that flooding protocols guarantee message relay (it is allowed that a message sent by the adversary is only received by a subset of honest parties), as this definition suffices for most blockchain protocols [GKL15, PS17a, DGKR18, PS18, CM19, YMR+19], since they do not depend on relaying of received messages at the flooding layer. To achieve message relay, one can simply let the honest parties re-distribute the received messages.

2.3 Additional Notation

We use the notation $\Gamma_s^{\lambda}(G)$ for the set of neighbors of a party (usually the sender) s at distance at most λ in a graph G. When clear from the context, we omit both s and G for this set and merely write Γ^{λ}. For two random variables X and Y, we will write $X \preceq Y$ if Y stochastically dominates X, i.e. $\Pr[Y \geq k] \geq \Pr[X \geq k]$ for all k.

2.4 Bounds

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

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

Lemma 2 (Convergence of Geometric Sum)**.** *Let $c, a \in \mathbb{R}$ and $|a| < 1$ then*

$$\sum_{n=0}^{\infty} c \cdot a^n = \frac{c}{1-a}.$$

Lemma 3. *Let $c, a \in \mathbb{R}$ s.t. $|a| < 1$ and $c \geq 2$ then*

$$\sum_{n=1}^{\infty} a^{(c^n)} \leq \frac{a}{1-a}. \tag{3}$$

Proof. First, note that

$$\sum_{n=1}^{\infty} a^{(c^n)} \leq \sum_{n=0}^{\infty} a^{(c^{n+1})} = \sum_{n=0}^{\infty} a^{c\cdot(c^n)}. \tag{4}$$

Now, note that for all $n \in \mathbb{N}$ we have that

$$1 + c^n \leq c \cdot c^n \implies a^{c \cdot c^n} \leq a^{1+c^n}. \tag{5}$$

Hence, by combining Equations (4) and (5) we get that

$$\sum_{n=1}^{\infty} a^{(c^n)} \leq \sum_{n=0}^{\infty} a \cdot a^{(c^n)}. \tag{6}$$

Now, Equation (3) follows by the convergence of the geometric sum (Lemma 2) and the fact that for all $n \in \mathbb{N}$ we have that $a^{(c^n)} \leq a^n$. □

3 Warm Up: Optimal Flooding With a Constant Diameter and Linear Neighborhood

In this section, we present our protocol ECCast and show that it is a flooding protocol with a maximum per-party communication of $O(l)$, a total communication complexity of $O(l \cdot n)$, and a diameter of 2.

Our protocol ECCast is parameterized by an erasure correcting code scheme that shares a message into n shares and a cryptographic accumulator. When a sender wishes to send, they will share the message into n shares and send a unique share to each party. When a party receives such a share, they will forward the share they receive to *all* other parties. This will ensure that each party ends up receiving as least as many shares as there are honest parties. Therefore, the only thing that can prevent honest parties from reconstructing the original message is if they try to reconstruct from some shares that were not sent by the original sender. To prevent this, we use the cryptographic accumulator.

Protocol $\mathsf{ECCast}(\zeta, \alpha)$

The protocol is parameterized by, a (n, e)-ECCS ζ for some $e \in \mathbb{N}$, and a cryptographic accumulator α. Each party $p_i \in \mathcal{P}$ keeps track of a set of shares received for a particular accumulator z, $\texttt{ReceivedShares}_i[z]$. Additionally, each party p_i keeps track of a set of received messages $\texttt{Received}_i$.

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

Send: When p_i receives $(\textit{Send}, m)$ they share the message m into shares $\zeta.\mathsf{Enc}(m) = s_1, \ldots, s_n$. Furthermore, they obtain an accumulated value and proofs for each share and its share number $z, \pi_1, \ldots, \pi_n = \alpha.\mathsf{Accumulate}(\{(s_j, j) \mid 1 \leq j \leq n\})$. For $1 \leq j \leq n$, the party now sends $(\textit{Forward}, s_j, j, \pi_j, z)$ to party p_j using the point-to-point channel between them. Finally, they add m to $\texttt{Received}_i$.

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

When party p_i receives a tuple (T, s, j, π, z) over a point-to-point channel where $\alpha.\mathsf{Verify}((s, j), \pi, z) = \top$ they add (s, j) to $\texttt{ReceivedShares}_i[z]$. Furthermore, p_i does the two following checks:

- If $|\texttt{ReceivedShares}_i[z]| \geq n - e$, then they
  1. Obtain a sequence of shares $s_1, \ldots, s_n$ by letting $s_j = s$ if $(s, j) \in \texttt{ReceivedShares}_i[z]$ and otherwise sets $s_j = \perp$ (i.e. if no such pair is in $\texttt{ReceivedShares}_i[z]$).
  2. Decode the shares and add the recovered message to the set of received messages, $\texttt{Received}_i \coloneqq \texttt{Received}_i \cup \{\zeta.\mathsf{Dec}(s_1, \ldots, s_n)\}$.

- If $T = \textit{Forward}$, it is the first time they receive (T, s, j, π, z), and $j = i$, then they send $(\textit{Receive}, s, j, \pi, z)$ to all parties over their respective point-to-point channels.

We now prove the following theorem.

Theorem 3. *Let $e \geq n \cdot (1-\gamma)$, let ζ be a (n, e)-ECCS, and let α be a WSCAS, then the protocol $\mathsf{ECCast}(\zeta, \alpha)$ is a strong $(2 \cdot \Delta_{NET})$-flooding protocol.*

Proof. Let m be a message that is input to some honest party s at time t, and let the accumulated value that is sent out be z. The delivery guarantees for the underlying point-to-point channels ensures that at latest at time $t + \Delta_{\text{NET}}$ any honest party p_i will have received a $(\textit{Forward}, s_i, i, r, \pi_i, z)$ s.t. $\alpha.\mathsf{Verify}((s, i), \pi_i, z) = \top$ (by correctness of the WSCAS). This implies that this is the latest point any honest party will forward $(\textit{Receive}, s_i, i, \pi, z)$ to all other parties. By the delivery guarantees of the point-to-point channels, these messages will be delivered at the latest at time $t + 2 \cdot \Delta_{\text{NET}}$. Hence, if p_i is an honest party, then the size of $\texttt{ReceivedShares}_i[z]$ will be at least $n \cdot \gamma = n - n \cdot (1 - \gamma) \geq n - e$. Therefore any honest party p_i will be able to reconstruct the original message at the latest at time $t + 2 \cdot \Delta_{\text{NET}}$ unless there are some tuple (s_j, j) and π where $\alpha.\mathsf{Verify}((s, j), \pi, z) = \top$ and where s_j is not equal to an original share sent out by the sender s. However, this does not happen unless the adversary is able to break the collision-freeness of the WSCAS. □

Communication complexity of ECCast. Let us now analyze the complexity of $\mathsf{ECCast}(\zeta, \alpha)$ (for ζ and α instantiated as suggested by Theorem 3) when a message of length l is input. The neighborhood of each party is n as all parties will talk to all other parties. The per-party communication is given by the size of the neighborhood times the size of the tuple sent over each point-to-point channel. As each tuple consists of a bit ($\textit{Forward}$ or $\textit{Receive}$), a share, a sequence number of the share, an accumulator proof, and an accumulated value, we have that the communication for each party is given by

$$n \cdot (1 + \zeta.\texttt{ShareSize} + \log(n) + \alpha.\texttt{ProofSize} + \alpha.\texttt{AccSize}).$$

If we instantiate the ECCS with Reed-Solomon codes, we get that the size of each share is bounded by $O(l \cdot (\gamma \cdot n)^{-1})$. For a constant fraction of honest parties, we, therefore, have that $\zeta.\texttt{ShareSize} = O(l \cdot n^{-1})$. And by using an efficient WSCAS α with size of accumulated value and proof $O(\kappa)$ (see Equation (2)), we get that the communication of each party is bounded by

$$O(l + n \cdot (\log(n) + \kappa)).$$

As all parties use this communication, the total communication complexity is bounded by

$$O(n \cdot (l + n \cdot (\log(n) + \kappa))).$$

This communication is optimal when $l = \Omega(n \cdot (\log(n) + \kappa))$. We further remark, that the constant that is multiplied to l is only γ^{-1} times the constant of the Reed-Solomon codes.

4 Optimal Flooding With Logarithmic Neighborhood and Diameter

In this section, we show how to obtain a flooding protocol with (asymptotically) optimal communication complexity. We achieve this in two steps. First, we define a weaker notion, denoted *weak flooding* and propose an instantiation of it. Then we show how to lift the security guarantees from a weak flooding protocol to achieve a full-fledged flooding protocol.

4.1 Weak Flooding

Informally, a weak flooding protocol is a flooding protocol that, instead of being guaranteed to deliver all messages to all parties, only ensures that there is a lower bound on the probability that each party receives a message. For the sake of simplicity, we first give an informal definition of a weak flooding protocol below.

Definition 4 (Weak Flooding (informal))**.** 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, parties may get a message as output. We say that Π is a *weak* (Δ, ξ)-flooding protocol if at any time t when a message m is input to some honest party, then it must be that for any $p_i \in \mathcal{H}$

$$\Pr[p_i \text{ receives } m \text{ at latest at time } t + \Delta] \geq \xi.$$

The reason that this is only an informal definition is that the definition fails to encapsulate what happens if multiple messages are sent in a protocol. For example, if a sequence of messages is sent at the same time, then a protocol satisfying the definition above could let all messages be delivered with probability ξ or deliver none of them with probability $(1 - \xi)$. Such a definition is, therefore, insufficient if one wishes to amplify the delivery guarantees for multiple messages. Therefore, in order to get a definition where it is reasonable to amplify the delivery probability, it is, therefore, necessary for us to describe how the delivery of several messages relates.

To formally define this property, we first introduce random variables for each delivery event in order to be able to exactly describe what kind of dependency is allowed between delivery events.

Definition 5 (Timely delivery)**.** We say that a message m input at time t is Δ-*timely-delivered* for a party p_i if p_i has output m at the latest at time $t + \Delta$. We let $\texttt{Timely}_{m,i}(\Delta)$ denote the induced indicator variable that is equal to 1 if the predicate holds and 0 otherwise.

Using this, we next define a weak flooding algorithm.

Definition 6 (Weak Flooding)**.** Let Π be a protocol executed by parties $\mathcal{P}$, where each party $p \in \mathcal{P}$ can input a *set* of messages at any time, and as a consequence, parties may get a set of messages as output. We say that Π is a *weak* (Δ, ξ)-flooding protocol if for any such set of messages $\{m_1, m_2, \ldots, m_\eta\}$ input to some honest party in the execution of Π where each message has never been sent over any channel before, then for any party $p_i \in \mathcal{H}$ there exist independent binary random variables $X_1, \ldots, X_\eta$ where for all $z \in \{1, 2, \ldots, \eta\}$, $\Pr[X_z = 1] \geq \xi$, and

$$\sum_{j=1}^{\eta} X_j \preceq \sum_{j=1}^{\eta} \texttt{Timely}_{m_j,i}(\Delta).$$

Remark 1. Directly requiring the `Timely` random variables to be independent would be a too strong requirement: An adversary can always ensure the delivery of a message if some other message is delivered by simply "injecting" the message on all point-to-point channels, thereby making the delivery events dependent. Note, however, that this influence can only be exerted in a way that increases the probability for delivery events to appear, which is what is captured in the definition above.

Protocol description. We now show that the protocol ERFlood[4] from [MNT22], actually is a weak flooding protocol, by providing a new analysis for the protocol. Concretely, we will make a

[4]The name ERFlood was given due to the relation to Erdős–Rényi-graphs.

different analysis and show that with just a constant expected degree ERFlood ensures that the probability that a party receives a message that is input for the first time is also constant.

For completeness, we first recap the protocol, which we phrase using the abstraction of *neighborhood selection algorithm* as in [LMM+22]. This abstraction allows our intermediate results to be used for different algorithms. However, unlike in their work, we consider only one neighborhood selection algorithm for the entire protocol instead of letting each party have a unique one.

Protocol $\mathsf{Flood}(\mathcal{N})$

The protocol is parameterized by a neighborhood selection algorithm $\mathcal{N}$. 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 neighbors $N \xleftarrow{\$} \mathcal{N}_p$ 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 $(\mathit{GetMessages})$ they return $\texttt{Relayed}_i$.

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

When talking about a set of messages input to a party (as required by the definition of a weak flooding protocol), we will simply interpret this as that the party which receives the set of messages will send each of them immediately after the other using the send command. The particular neighborhood selection algorithm which we will concentrate on in this section is the one where each party simply selects all other parties as their neighborhood with an independent probability ρ. We denote this algorithm with ERS, which is shorthand for *Erdős–Rényi selection*.

Algorithm $\mathsf{ERS}(\rho)$

Let $N \coloneqq \varnothing$.
Let $P \coloneqq \mathcal{P}$.
while $P \neq \varnothing$ **do**
    Pick $r \in P$.
    Sample $c \xleftarrow{\$} \mathcal{U}([0,1])$.
    if $c \leq \rho$ **then**
        Update $N \coloneqq N \cup \{r\}$.
    Update $P \coloneqq P \setminus \{r\}$.
return N.

Remark 2. [LMM+22] showed that this sampling can be done more efficiently by first sampling a number k from the binomial distribution with parameters n and ρ, and afterward, sample k parties from $\mathcal{P}$ without replacement.

Using this abstraction, we denote the protocol $\mathsf{ERFlood}(\rho) \coloneqq \mathsf{Flood}(\mathsf{ERS}(\rho))$ and prove the following theorem in Section 4.1.1. Note that we explicitly quantify over the number of parties n after the existential quantification of the success probability bound, to highlight that the probability is independent of the number of parties.

Theorem 4. *There exists $\xi \in (0,1]$ s.t. for any $n \geq 11 \cdot \gamma^{-1}$ there is a $\rho = O(n^{-1})$ and $\Delta = O(\log(n) \cdot \Delta_{NET})$ s.t. the protocol $\mathsf{ERFlood}(\rho)$ is a weak (Δ, ξ)-flooding protocol.*

Previous analysis [MNT22] of this protocol only showed the necessary condition that $\rho = \Omega(\log(n) \cdot n^{-1})$ so that the overall protocol delivers the message to all parties with constant probability. Our analysis instead proves that $\rho = O(n^{-1})$ is enough to guarantee that with constant probability of success, any fixed party receives the message.

4.1.1 Security Analysis of ERFlood

In order to prove that ERFlood is a weak flooding protocol, we have to prove that for any party, the probability that this party receives a specific message is constant. We do so by re-using the idea of the honest sending process from [LMM+22] and the probability of a timely delivery to the probability that a party has an incoming edge in this graph. The idea of the honest sending process, which we recap below, is to let it mimic the sending of a particular message where only the honest parties participate in the distribution of the message, and the adversary only delivers the message on the point-to-point channels at the latest point in time possible. For completeness, we provide the definition of the honest sending process from [LMM+22] below.

Definition 7 (The honest sending process)**.** Let $s \in \mathcal{H}$ be an honest party, let $\mathcal{N}$ be a family of neighborhood selection algorithms, and let $\lambda \in \mathbb{N}$ be a distance. We let the *honest sending process*, $\mathsf{HSP}(s, \mathcal{N}, \lambda)$, 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 $\texttt{ToFlip} := \{(s, 0)\}$ of nodes and their distance from p that are to have their edges decided.
2. The process proceeds with taking out the first element of $\texttt{ToFlip}$ until $\texttt{ToFlip} == \varnothing$. Let the element that is taken out out of $\texttt{ToFlip}$ be denoted by (p', i) and do the following:
   (a) Let $N \xleftarrow{\$} \mathcal{N}$ and set $N := N \cap \mathcal{H}$.
   (b) Update the set of edges $E := E \cup \{(p', p'') \mid p'' \in N\}$ and let $\texttt{Flipped} := \texttt{Flipped} \cup \{p'\}$.
   (c) If $i + 1 < \lambda$, for all $p'' \in N \cap \mathcal{H} \setminus (\texttt{Flipped} \cup \texttt{ToFlip})$ add $(p'', i + 1)$ to $\texttt{ToFlip}$.
3. Finally, return $G = (\mathcal{H}, E)$.

Similarly to [LMM+22], we now relate the probability of an event in the honest sending process (in our case that a party is in the neighborhood of the graph produced by the honest sending process) to the probability of a timely delivery and use this to show that ERFlood is a weak flooding protocol for certain parameters.

Lemma 4. *Let $\lambda \in \mathbb{N}$ be a distance, let $\rho \in [0,1]$ and let $\Delta := \lambda \cdot \Delta_{NET}$. Further, let $s_{min} \in \mathcal{H}$ and $p_{min} \in \mathcal{H}$ s.t. when $G \xleftarrow{\$} \mathsf{HSP}(s_{min}, \mathsf{ERS}(\rho), \lambda)$ then $\Pr[p_{min} \in \Gamma^{\lambda}_{s_{min}}(G)]$ is minimized over all such $s, p \in \mathcal{H}$. If*

$$\xi \leq \Pr[p_{min} \in \Gamma^{\lambda}_{s_{min}}(G)],$$

then $\mathsf{ERFlood}(\rho)$ is a weak (Δ, ξ)-flooding protocol.

Proof. Let $m_1, m_2, \ldots, m_\eta$ be the set of "fresh" (meaning that they have not been sent over any channel before) messages input to some honest party s in the execution of $\mathsf{ERFlood}(\rho)$ at time t, and let p_i be an honest party. Now, we let $G_1 \xleftarrow{\$} \mathsf{HSP}(s, \mathsf{ERS}(\rho), \lambda), \ldots, G_\eta \xleftarrow{\$} \mathsf{HSP}(s, \mathsf{ERS}(\rho), \lambda)$

and let $T_1, \ldots, T_\eta$ be indicator variables such that T_j indicates if the event $p_i \in \Gamma_s^\lambda(G_j)$ happened. It is clear that $T_1, \ldots, T_\eta$ are independent. Furthermore, we have for any T_j that

$$\Pr[T_j = 1] = \Pr[p_i \in \Gamma_s^\lambda(G_j)] \geq \Pr[p_{\min} \in \Gamma_{s_{\min}}^\lambda(G)] \geq \xi.$$

To prove that $\mathsf{ERFlood}(\rho)$ is weak (Δ, ξ)-flooding protocol it is thus left to show that

$$\sum_{j=1}^{\eta} T_j \preceq \sum_{j=1}^{\eta} \mathtt{Timely}_{m_j,i}.$$

To do so, we follow the proof of [LMM+22, Lemma 5 on p. 16] by coupling the execution of $\mathsf{ERFlood}(\rho)$ to the creation of a graph from the honest sending process. However, instead of creating just one graph, we observe the execution and create multiple graphs $G'_1, \ldots, G'_\eta$. For all G_j, we let $G'_j = (\mathcal{V}'_j, E'_j)$ and let the set of nodes be all honest parties (i.e., $\mathcal{V}'_j = \mathcal{H}$). We add an edge between to honest parties (p_k, p_z) to E'_j if and only if p_k sent the message m_j to p_z and p_z receives it before time $t + \lambda \cdot \Delta_{\text{NET}}$. With this definition it is clear that if p_i has an incoming edge in G'_j then $\mathtt{Timely}_{m_j,i} = 1$. We now do a coupling between the graphs $G'_1, \ldots, G'_\eta$ and $G_1, \ldots G_\eta$ by first running the execution of $\mathsf{ERFlood}(\rho)$ and then define a new graphs $\widetilde{G_1}, \ldots, \widetilde{G_\eta}$ where each graph $\widetilde{G_j}$ will be defined in terms of G'_j. We define $\widetilde{G_j} = (\widetilde{\mathcal{V}_j}, \widetilde{E_j})$ by letting $\widetilde{\mathcal{V}_j} = \mathcal{H}$ and define $\widetilde{E_j}$ by duplicating the edges from E'_j to $\widetilde{E_j}$ for all parties within distance $\lambda - 1$ of s in G'_j.

Now, let $\widetilde{T_1}, \ldots, \widetilde{T_\eta}$ be indicator variables such that $\widetilde{T_j}$ indicates if the event $p_i \in \Gamma_s^\lambda(\widetilde{G_j})$ happened. Observe, that if a $\widetilde{T_j} = 1$ then party p_i has an incoming edge in $\widetilde{G_j}$. This implies that party also has an edge in G'_j as $\widetilde{E_j} \subseteq E'_j$ which again (as argued before) implies that $\mathtt{Timely}_{m_j,i} = 1$. Therefore, it is clear that

$$\sum_{j=1}^{\eta} \widetilde{T_j} \leq \sum_{j=1}^{\eta} \mathtt{Timely}_{m_j,i},$$

which again implies that for any $c \in \mathbb{R}$ we have that

$$\Pr\left[\sum_{j=1}^{\eta} \mathtt{Timely}_{m_j,i} \leq c\right] \leq \Pr\left[\sum_{j=1}^{\eta} \widetilde{T_j} \leq c\right].$$

It is thus left to show that $\sum_{j=1}^{\eta} T_j \sim \sum_{j=1}^{\eta} \widetilde{T_j}$. We will do so by arguing that for any G_j we have that $G_j \sim \widetilde{G_j}$ and that $\widetilde{G_j}$ independent. Note that each honest party selects their neighbors in $\mathsf{ERFlood}(\rho)$ using the same neighborhood selection as used in the honest sending process. Furthermore, each honest party makes independent draws of neighbors for each different message. By assumption, each message is different, and this, therefore, ensures that the distributions of the edges in $\widetilde{G_j}$ are really independent. It is, therefore, only left to argue that there will not be any parties within distance $\lambda - 1$ in any graph G'_j that have not had their edges selected. This follows directly by the assumption that the delay on the point-to-point channels is at most Δ_{NET} and therefore, a party at a distance k in G'_j will at latest select their neighbors $k \cdot \Delta_{\text{NET}}$ time after a message is input to the initial sender. □

Lemma 5. *Let $\delta_1, \delta_2, \alpha \in [0, 1]$. Further, let $\phi \in \mathbb{R}$ be an expected expansion factor, $d \in \mathbb{R}$ be a constant, and let $\rho \coloneqq \frac{d}{h}$, let $\lambda \coloneqq \frac{\log\left(\frac{\alpha \cdot |\mathcal{H}|}{(1-\delta_1)\cdot d}\right)}{\log((1-\delta_2)\cdot\phi)}$. Finally, let $s \in \mathcal{H}$ and let $G \xleftarrow{\$} \mathsf{HSP}(s, \mathsf{ERS}(\rho), \lambda)$. If*

$$e^{-d\cdot\alpha} + \frac{\alpha \cdot \phi}{1 - \alpha} \leq 1 \tag{7}$$

and

$$(1-\delta_2)\cdot\phi \geq 2, \tag{8}$$

then for any party $p \in \mathcal{H}$

$$\frac{\alpha\cdot|\mathcal{H}|-1}{|\mathcal{H}|}\cdot\left(1-e^{-\frac{\delta_1^2\cdot d}{2}}-\frac{e^{-\frac{\delta_2^2\cdot(1-\delta_1)\cdot d}{2}}}{1-e^{-\frac{\delta_2^2\cdot(1-\delta_1)\cdot d}{2}}}\right) \leq \Pr[p\in\Gamma_s^\lambda(G)].$$

Proof. We let Γ^λ be the set of neighbors of the sender s at a distance at most λ in a graph G. Using the law of total probability, we note that the probability that p is in the close neighborhood of the sender is lower bounded by the probability that the sender has a large honest neighborhood $\alpha\cdot|\mathcal{H}| \leq |\Gamma^\lambda|$ for some $\alpha\in(0,1]$ *and* the party p is in fact in this neighborhood, i.e., $p\in\Gamma^\lambda$:

$$\begin{aligned}
&\Pr\left[p\in\Gamma^\lambda\right] = \Pr\left[p\in\Gamma^\lambda\cap\alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\right] + \Pr\left[p\in\Gamma^\lambda\cap\alpha\cdot|\mathcal{H}|>|\Gamma^\lambda|\right]\\
&\geq \Pr\left[p\in\Gamma^\lambda\cap\alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\right]\\
&= \Pr\left[p\in\Gamma^\lambda \mid \alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\right]\cdot\Pr\left[\alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\right].
\end{aligned} \tag{9}$$

We now bound these two probabilities individually.

Let us first bound the probability that party p is in the set of neighbors i.e., $p\in\Gamma^\lambda$ given the set of neighbors is "large", $\alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|$. For this, we use the that all honest parties have an equal probability of appearing inside Γ^λ (except the sender, who is always there) and the law of total probability.

$$\begin{aligned}
&\Pr\left[p\in\Gamma^\lambda \mid \alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\right]\\
&= \sum_{a=1}^{|\mathcal{H}|}\Pr\left[p\in\Gamma^\lambda \mid \alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\cap|\Gamma^\lambda|=a\right]\cdot\Pr\left[|\Gamma^\lambda|=a \mid \alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\right]\\
&= \sum_{a=\alpha\cdot|\mathcal{H}|}^{|\mathcal{H}|}\Pr\left[p\in\Gamma^\lambda \mid |\Gamma^\lambda|=a\right]\cdot\Pr\left[|\Gamma^\lambda|=a \mid \alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\right]\\
&= \sum_{a=\alpha\cdot|\mathcal{H}|}^{|\mathcal{H}|}\frac{a-1}{|\mathcal{H}|}\cdot\Pr\left[|\Gamma^\lambda|=a \mid \alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\right]\\
&\geq \frac{\alpha\cdot|\mathcal{H}|-1}{|\mathcal{H}|}\cdot\sum_{a=\alpha\cdot|\mathcal{H}|}^{|\mathcal{H}|}\Pr\left[|\Gamma^\lambda|=a \mid \alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\right]\\
&= \frac{\alpha\cdot|\mathcal{H}|-1}{|\mathcal{H}|}.
\end{aligned} \tag{10}$$

Let us now bound the probability that the neighborhood of the sender is "large," i.e., the event $\alpha\cdot|\mathcal{H}|\geq|\Gamma^\lambda|$. To do so, we reuse parts of a proof by Matt et al. [MNT22, Lemma 3 on p. 23-25], but look at a more specific setting where and directly insert the parameters for this setting. Further, we only reuse parts of the proof as we are only interested in bounding the neighborhood of the sender and not the neighborhoods of all parties.

We define $D \coloneqq (|\Gamma^\lambda| < \alpha\cdot|\mathcal{H}|)$, i.e., the event that the sender does not reach at least a α-fraction of the minimum honest nodes within λ steps. We have that

$$\Pr\left[\alpha\cdot|\mathcal{H}|\leq|\Gamma^\lambda|\right] = 1-\Pr[D],$$

and hence it is sufficient to bound $\Pr[D]$. We let θ^ν be the set of honest parties that are at exactly distance ν from the sender, and define the parties at exactly distance 0 to be the sender,

$\theta^0 \triangleq \{s\}$. We now define a series of events. First, we define the event that the number of immediate neighbors of the sender deviates significantly from the mean

$$A_0 \coloneqq (|\theta^1| > (1-\delta_1)\cdot d).$$

Next, we define the events that the number of neighbors in $\nu+1$ is at least $(1-\delta_2)\cdot\phi$ times the number of neighbors at step ν

$$B_\nu \coloneqq \left(|\theta^{\nu+1}| > (1-\delta_2)\cdot\phi\cdot|\theta^\nu|\right).$$

Finally, we define the event that the neighbors within distance ν is at least $\alpha\cdot|\mathcal{H}|$

$$C_\nu \coloneqq (|\Gamma^\nu| \geq \alpha\cdot|\mathcal{H}|).$$

As convenient notation we let $A_\nu \coloneqq B_\nu \vee C_\nu$ for $\nu = 1,\ldots,\lambda-1$. Now, note that

$$\lambda = \frac{\log\left(\frac{\alpha\cdot|\mathcal{H}|}{(1-\delta_1)\cdot d}\right)}{\log((1-\delta_2)\cdot\phi)} + 1 \quad\iff\quad (1-\delta_1)\cdot d\cdot((1-\delta_2)\cdot\phi)^{\lambda-1} = \alpha\cdot|\mathcal{H}|.$$

Therefore, if A_0 and $B_1,\ldots,B_\lambda$ holds, then

$$\begin{aligned}
|\Gamma^\lambda| &= \sum_{\nu=0}^{\lambda}|\theta^\nu| \\
&= 1 + \sum_{\nu=0}^{\lambda-1}|\theta^{\nu+1}| \\
&\geq 1 + \sum_{\nu=0}^{\lambda-1}((1-\delta_2)\cdot\phi)^\nu\cdot|\theta^1| \\
&\geq 1 + (1-\delta_1)\cdot d\cdot\sum_{\nu=0}^{\lambda-1}((1-\delta_2)\cdot\phi)^\nu \\
&> (1-\delta_1)\cdot d\cdot((1-\delta_2)\phi)^{\lambda-1} \\
&= \alpha\cdot|\mathcal{H}|.
\end{aligned}$$

Similarly, if just some C_ν holds then we get that $|\Gamma^\lambda| \geq \alpha\cdot|\mathcal{H}|$. Therefore, by contraposition, we have that

$$\left(\bigwedge_{\nu=0}^{\lambda-1} A_\nu \implies \neg D\right) \quad\iff\quad \left(D \implies \bigvee_{\nu=0}^{\lambda-1}\neg A_\nu\right).$$

Hence, we get the following:

$$\begin{aligned}
\Pr[D] &\leq \Pr\left[\bigcup_{i=0}^{\lambda-1} \neg A_i\right] \\
&\leq \sum_{i=0}^{\lambda-1} \Pr\left[\neg A_i \mid \bigcap_{j<i} A_j\right] \\
&= \Pr[\neg A_0] + \sum_{i=1}^{\lambda-1} \Pr\left[\neg A_i \mid \bigcap_{j<i} A_j\right] \\
&= \Pr[\neg A_0] + \sum_{i=1}^{\lambda-1} \Pr\left[\neg B_i \cap \neg C_i \mid \bigcap_{j<i} A_j\right] \\
&\leq \Pr[\neg A_0] + \sum_{i=1}^{\lambda-1} \Pr\left[\neg B_i \mid \bigcap_{j<i} A_j \cap \neg C_i\right] \\
&= \Pr[\neg A_0] + \sum_{i=1}^{\lambda-1} \Pr\left[\neg B_i \mid \bigcap_{1\leq j<i} B_j \cap \neg C_i \cap A_0\right].
\end{aligned} \tag{11}$$

We now state and prove a bound on the individual probabilities inside the sum.

Claim 1 (Fast expansion to small fraction)**.** *For any $i \in \{1, \ldots, \lambda - 1\}$ we have*

$$\Pr\left[\neg B_i \mid \bigcap_{1\leq j<i} B_j \cap \neg C_i \cap A_0\right] \leq e^{-\frac{\delta_2^2 \cdot ((1-\delta_2)\cdot\phi)^\nu \cdot (1-\delta_1)\cdot d}{2}}.$$

Proof. We look at the probability space where $\bigcap_{1\leq j<i} B_j \cap \neg C_i \cap A_0$ holds. We define the number of parties that are reachable at a distance ν to be $r \coloneqq |\theta^\nu|$ and let the number of honest parties that have not been reached so far by $U \coloneqq \mathcal{H} \setminus \Gamma^\nu$. For each $u \in U$ we introduce an indicator variable X_u which indicates if u is in $\theta^{\nu+1}$. As the probability that there is an edge between any two honest parties is independent of other edges, we have

$$\Pr[X_u = 1] = 1 - (1-\rho)^r \geq 1 - e^{-\rho r}.$$

The size of $\theta^{\nu+1}$ is the sum of these independent variables, i.e.,

$$|\theta^{\nu+1}| = \sum_{u\in U} X_u.$$

As we are looking at the case where $\neg C_i$, we have $|U| \geq (1-\alpha)\cdot|\mathcal{H}|$ which by linearity of expectations gives us that

$$\mathrm{E}[|\theta^{\nu+1}|] \geq |\mathcal{H}| \cdot (1-\alpha) \cdot \left(1 - e^{-\rho r}\right).$$

We now subtract $\phi \cdot r$ on each side of the inequality above and insert $\rho = \frac{d}{h} \leq \frac{d}{|\mathcal{H}|}$, in order to show that the expected value is larger than $\phi \cdot r$. We get

$$\begin{aligned}
\mathrm{E}[|\theta^{\nu+1}|] - \phi\cdot r &\geq |\mathcal{H}| \cdot (1-\alpha) \cdot \left(1 - e^{-\rho r} - \frac{\phi \cdot r}{|\mathcal{H}| \cdot (1-\alpha)}\right) \\
&= |\mathcal{H}| \cdot (1-\alpha) \cdot \left(1 - e^{-d\frac{r}{|\mathcal{H}|}} - \frac{\phi \cdot r}{|\mathcal{H}| \cdot (1-\alpha)}\right).
\end{aligned}$$

We let $x = \frac{r}{|\mathcal{H}|}$ and set $f(x) = 1 - e^{-dx} - x\frac{\phi}{(1-\alpha)}$. We differentiate this twice and find $f''(x) = -d^2 e^{-dx} \leq 0$, which implies that f is concave, which again implies that the minimum values are at one of the endpoints of the function. As $x \in [0, \alpha]$ it is enough to check that $f(0) \geq 0$ and $f(\alpha) \geq 0$ which will imply that $\mathrm{E}[|\theta^{\nu+1}|] \geq \phi \cdot |\theta^{\nu}|$.

$$f(0) = 1 - e^{-d\cdot 0} - 0 = 0.$$
$$f(\alpha) = 1 - e^{-d\cdot\alpha} - \frac{\alpha\cdot\phi}{1-\alpha} \geq 0 \quad \Longleftrightarrow \quad e^{-d\cdot\alpha} + \frac{\alpha\cdot\phi}{1-\alpha} \leq 1.$$

We now use Chernoff (Lemma 1) to bound the probability that this is not the case which means that for any $\delta_2 \in [0, 1]$, we get that

$$\Pr[|\theta^{\nu+1}| \leq (1-\delta_2)\cdot\phi\cdot|\theta^{\nu}|] \leq e^{-\frac{\delta^2\cdot\phi\cdot|\theta^{\nu}|}{2}}$$

However, $\bigcap_{j<i} B_j \cap A_0$ and $(1-\delta_2)\cdot\phi \geq 1$ ensures that

$$|\theta^{\nu}| \geq ((1-\delta_2)\cdot\phi)^{\nu}\cdot(1-\delta_1)\cdot d.$$

Hence, within the probability space where $\bigcap_{1\leq j<i} B_j \cap \neg C_i \cap A_0$ holds we have that

$$\Pr[|\theta^{\nu+1}| \leq (1-\delta_2)\cdot\phi\cdot|\theta^{\nu}|] \leq e^{-\frac{\delta_2^2\cdot((1-\delta_2)\cdot\phi)^{\nu}\cdot(1-\delta_1)\cdot d}{2}}.$$

□

We apply Chernoff for bounding the probability of the event A_0, the claim we have just proven, and Lemma 3 to the bound we established in Equation (11) to obtain a final bound for the probability of the event D

$$\begin{aligned}
\Pr[D] &\leq e^{-\frac{\delta_1^2\cdot d}{2}} + \sum_{\nu=1}^{\lambda-1} e^{-\frac{\delta_2^2\cdot((1-\delta_2)\cdot\phi)^{\nu}\cdot(1-\delta_1)\cdot d}{2}} \\
&< e^{-\frac{\delta_1^2\cdot d}{2}} + \sum_{\nu=1}^{\infty} e^{-\frac{\delta_2^2\cdot((1-\delta_2)\cdot\phi)^{\nu}\cdot(1-\delta_1)\cdot d}{2}} \\
&= e^{-\frac{\delta_1^2\cdot d}{2}} + \sum_{\nu=1}^{\infty} \left(e^{-\frac{\delta_2^2\cdot(1-\delta_1)\cdot d}{2}}\right)^{(((1-\delta_2)\cdot\phi)^{\nu})} \\
&\leq e^{-\frac{\delta_1^2\cdot d}{2}} + \frac{e^{-\frac{\delta_2^2\cdot(1-\delta_1)\cdot d}{2}}}{1 - e^{-\frac{\delta_2^2\cdot(1-\delta_1)\cdot d}{2}}}.
\end{aligned}$$

Hence, it follows that

$$\Pr\left[\alpha\cdot|\mathcal{H}| \leq |\Gamma^{\lambda}|\right] \geq 1 - e^{-\frac{\delta_1^2\cdot d}{2}} - \frac{e^{-\frac{\delta_2^2\cdot(1-\delta_1)\cdot d}{2}}}{1 - e^{-\frac{\delta_2^2\cdot(1-\delta_1)\cdot d}{2}}}. \tag{12}$$

The desired bound on the probability for late delivery now follows from Equations (9), (10) and (12).

□

Using Lemmas 4 and 5, we are now ready to prove our main theorem for ERFlood. For completeness we restate it below.

Theorem 4. *There exists $\xi \in (0,1]$ s.t. for any $n \geq 11 \cdot \gamma^{-1}$ there is a $\rho = O(n^{-1})$ and $\Delta = O(\log(n) \cdot \Delta_{NET})$ s.t. the protocol $\mathsf{ERFlood}(\rho)$ is a weak (Δ, ξ)-flooding protocol.*

Proof. We let $\delta_1 := \delta_2 := \alpha := \frac{1}{10}$ and let $\phi := \frac{20}{9}$. This ensures that Equation (8) is always fulfilled and if we let

$$d \geq -10 \cdot \ln\left(\frac{61}{81}\right)$$

then also Equation (7) is fulfilled. Hence, if we let $\rho = \frac{d}{h}$ then Lemma 5 ensures that for $\lambda = O(\log(n))$ then for any sender s and receiver p (in particular also the sender and receiver that minimizes the probability for the event below) then if $G \xleftarrow{\$} \mathsf{HSP}(s, \mathsf{ERS}(\rho), \lambda)$ then

$$\frac{h-10}{10 \cdot h} \cdot \left(1 - e^{-\frac{d}{200}} - \frac{e^{-\frac{d}{180}}}{1 - e^{-\frac{d}{180}}}\right) \leq \Pr[p \in \Gamma_s^{\lambda}(G)].$$

We note that $n \geq 11 \cdot \gamma^{-1} \implies h \geq 11$ and therefore that

$$\frac{h-10}{10 \cdot h} \geq \frac{1}{110}.$$

Hence, for large enough (but still constant) d then $\xi := \frac{1}{110} \cdot \left(1 - e^{-\frac{d}{200}} - \frac{e^{-\frac{d}{180}}}{1-e^{-\frac{d}{180}}}\right) \in (0,1]$ is a constant independent of n. Lemma 4 implies that $\mathsf{ERFlood}(\rho)$ is a weak (Δ, ξ)-flooding protocol with $\Delta = \Delta_{\text{NET}} \cdot \lambda = O(\log(n) \cdot \Delta_{\text{NET}})$. Further, note that this holds when $\rho = \frac{d}{h} = O(\gamma^{-1} \cdot n^{-1}) = O(n^{-1})$, where the last equality follows because we assume that a constant fraction of the parties is honest. □

4.2 Flooding Amplification

We present a compiler that amplifies delivery guarantees of a weak flooding protocol to full fledge flooding. The protocol $\mathsf{WeakFlood2Flood}$ is parameterized by a *weak* flooding protocol, an erasure correcting code scheme (ECCS), and a cryptographic accumulator. The idea of the protocol is that when a sender wishes to send a message, they divide the message into multiple shares using the ECCS.The sender will then send each of these shares using the weak flooding protocol. Each receiver will receive a set of shares and try to reconstruct the original message from this. Intuitively, if everybody receives sufficiently many of the original shares within the given time, then the only thing that can prevent an honest party from reconstructing the message sent by the sender is if an adversary managed to inject some "false shares" into the set of shares an honest party tries to reconstruct their message from. To prevent this from happening, the sender will create an accumulated value of all shares, and then instead of sending out only the share, they will send out the share, a proof, an accumulated value, and a proof that this share belongs to this accumulated value. On the receiving end, honest parties will group shares by the accumulated value they belong to and only try to reconstruct from shares that belong to the same accumulated value. Hence, an adversary will have to break the collision-free property of the accumulator scheme in order to inject such false shares.

It is, therefore, only left to ensure that all parties receive sufficiently many of the original shares. We will ensure this by instantiating $\mathsf{WeakFlood2Flood}$ with a weak flooding protocol $\mathsf{ERFlood}$, which we will instantiate such that each party is guaranteed to receive a constant fraction of the shares if a message is split into sufficiently many shares and set the parameters of the ECCS accordingly.

The formal description of the protocol can be found below.

Protocol WeakFlood2Flood(Π, ζ, α)

The protocol is parameterized by a weak flooding protocol Π, a (μ, e)-ECCS ζ for some $\mu, e \in \mathbb{N}$, and a cryptographic accumulator α.
Each party $p_i \in \mathcal{P}$ keeps track of a set of shares received for a particular accumulator z, $\texttt{ReceivedShares}_i[z]$. Additionally, each party p_i keeps track of a set of received messages $\texttt{Received}_i$.

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

Send: When p_i receives (Send, m) they share the message m into shares $\zeta.\mathsf{Enc}(m) = s_1, \ldots, s_\mu$. Furthermore, they obtain an accumulated value and proofs for each share and its share number $z, \pi_1, \ldots, \pi_\mu = \alpha.\mathsf{Accumulate}(\{(s_i, i) \mid 1 \leq i \leq \mu\})$. The party now draws a uniformly sampled random number $r \xleftarrow{\$} \mathcal{U}(\{0,1\}^\kappa)$. Now, the party inputs the set of messages $\{(s_j, j, r, \pi_j, z) \mid 1 \leq j \leq \mu\}$ to Π. Finally, they add m to $\texttt{Received}_i$.

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

When party p_i receives a tuple (s, j, r, π, z) in Π where $\alpha.\mathsf{Verify}((s, j), \pi, z) = \top$ they add (s, j) to $\texttt{ReceivedShares}_i[z]$. Furthermore, p_i check if $|\texttt{ReceivedShares}_i[z]| \geq \mu - e$. If that is the case p_i does the following:

1. Obtains a sequence of shares $s_1, \ldots, s_\mu$ by letting $s_j = s$ if $(s, j) \in \texttt{ReceivedShares}_i[z]$ and otherwise sets $s_j = \perp$ if no such pair is in $\texttt{ReceivedShares}_i[z]$.
2. Decode the shares and add the recovered message to the set of received messages, $\texttt{Received}_i \coloneqq \texttt{Received}_i \cup \{\zeta.\mathsf{Dec}(s_1, \ldots, s_\mu)\}$.

At first, it might seem strange that a sender samples a random number and attaches this to each share. The reason for this is that the weak flooding protocol we will instantiate WeakFlood2Flood with will only ensure that sufficiently many shares are delivered if they are sent for the first time in the execution of the protocol. Hence, to ensure that the probability that this happens is negligible in κ, we attach the random number. We note that the probability that an adversary manages to do so throughout the execution is bounded by the birthday paradox. For clarity of presentation, we will not propagate this exact probability through our proofs but merely assume that an adversary does not guess such a random number that an honest party will later attach to their message. Similarly, we will also assume that the adversary that no collision happens for the cryptographic accumulator.

Security of WeakFlood2Flood. We now state and prove the security of WeakFlood2Flood, given that the protocol is instantiated with a weak flooding protocol.

Theorem 5. *Let $\xi \in (0, 1]$, let $\Delta \in \mathbb{N}$, and let Π be a weak (Δ, ξ)-flooding protocol. Further, let $\delta \in (0, 1]$, let $\mu \in \mathbb{N}$, let $e \geq \mu \cdot (1 - (1 - \delta) \cdot \xi)$, let ζ be a (μ, e)-ECCS, and let α be a WSCAS. The probability that a message sent in the protocol WeakFlood2Flood(Π, ζ, α) is not delivered within Δ to all parties is less than*

$$|\mathcal{H}| \cdot e^{-\frac{\delta^2 \cdot \xi \cdot \mu}{2}}.$$

Proof. Let us look at a particular message that is sent by an honest sender s at time t by flooding the shares $s_1, \ldots, s_\mu$. We note that if a party p receives more than $(1 - \delta) \cdot \xi \cdot \mu$ number of shares,

then by properties of the (μ, e)-ECCS, then p is able to reconstruct the message unless p received some shares different from the original shares and tries to reconstruct from these. However, if this happens, this means that an honest party added a share and share number (s, i) to the set of shares for a particular accumulator, where (s, i) was not a part of the original shares used to construct the accumulated value, and hence this only happens with the same probability as a collision in the accumulation scheme happens. It is, therefore, sufficient to bound the probability that there exists a party that does not receive $(1-\delta)\cdot\xi\cdot\mu$ shares from the sender.

For each honest party $p_i \in \mathcal{H}$ we introduce random indicator variables $S_{i,1}, S_{i,2}, \ldots, S_{i,\mu}$ where $S_{i,j}$ indicates whether or not party p_i received share s_j before time $t + \lambda \cdot \Delta_{\text{NET}}$. Further, we introduce a variable that denotes how many shares party p_i receives from honest parties $S_i = \sum_{j=1}^{\mu} S_{i,j}$ at latest at time $\lambda \cdot \Delta_{\text{NET}}$. Because Π is a weak (ξ, Δ)-flooding protocol and the shares are now input to Π as messages for the first time, we know that there exists a sequence of independent variables $S^*_{i,1}, S^*_{i,2}, \ldots, S^*_{i,\mu}$ where $\Pr[S^*_{i,j} = 1] \geq \xi$. Furthermore, if we let their sum be given by $S^*_i = \sum_{j=1}^{\mu} S^*_{i,j}$ then we know that that S_i stochastically dominates S^*_i. That is that for any $k \in \mathbb{R}$ we have

$$\Pr[S_i \leq k] \leq \Pr[S^*_i \leq k].$$

Hence, it is sufficient to bound the probability for the event that $S^*_i \leq (1-\delta)\cdot\xi\cdot\mu$ in order to bound the probability that party p_i receives less than $(1-\delta)\cdot\xi\cdot\mu$ shares before $t + \Delta$. By linearity of expectation, we have that

$$\mathrm{E}[S^*_i] \geq \xi \cdot \mu.$$

Chernoff (Lemma 1) gives us that for any $\delta \in [0, 1)$ we have

$$\Pr\left[S^*_i \leq (1-\delta)\cdot\xi\cdot\mu\right] \leq e^{-\frac{\delta^2_3\cdot\xi\cdot\mu}{2}}.$$

Finally, we calculate the probability that there *exists any honest party* that does not receive at least $(1-\delta)\cdot\xi\cdot\mu$ shares using the union bound:

$$\begin{aligned}\Pr\left[\exists p_i \in \mathcal{H}, \text{ s.t. } p_i \text{ receives less than } (1-\delta)\cdot\xi\cdot\mu \text{ shares}\right] &\leq \sum_{p_i\in\mathcal{H}} \Pr\left[S_i \leq (1-\delta)\cdot\xi\cdot\mu\right] \\ &\leq |\mathcal{H}| \cdot e^{-\frac{\delta^2_3\cdot\xi\cdot\mu}{2}}.\end{aligned}$$

□

It is noteworthy that WeakFlood2Flood inherits the delivery guarantee of the weak flooding protocol that it is instantiated with. Next, we state that a direct corollary of the above theorem, namely that for certain parameters, WeakFlood2Flood is a strong flooding protocol.

Corollary 1. *Let $\xi \in (0, 1]$, let $\Delta \in \mathbb{N}$, and let Π be a weak (Δ, ξ)-flooding protocol. Further, let $\mu \geq \frac{\log(n)+\kappa}{\xi}$, let $e \geq \mu \cdot \left(1 - \frac{\xi}{2}\right)$, let ζ be a (μ, e)-ECCS, and let α be a WSCAS. The protocol WeakFlood2Flood(Π, ζ, α) is a strong Δ-flooding protocol.*

Proof. We note that $|\mathcal{H}| \leq n$ and use Theorem 5 instantiated with $\delta \coloneqq \frac{1}{2}$. □

4.3 Putting It All Together

We consider the instantiation ECFlood (from "erasure coded flood"), defined as ECFlood$(\rho, \zeta, \alpha) \coloneqq$ WeakFlood2Flood(ERFlood$(\rho), \zeta, \alpha)$, where the protocol compiler WeakFlood2Flood is instantiated with the weak flooding ERFlood(ρ) for $\rho \in (0, 1]$, ζ is a (μ, e)-ECCS and α is WSCAS scheme.

Communication complexity analysis. Consider the case where a single message of length l is input to an honest party, and let S denote the total number of messages sent by honest parties. Then, the total communication complexity of $\mathsf{ECFlood}(\rho, \zeta, \alpha)$ is upper bounded by S times the size of each of the messages sent for each share.

Each message consists of a share, the sequence number of the share, nonce of length κ, an accumulator proof, and an accumulated value. Hence, the total communication complexity will be upper bounded by

$$S \cdot (\zeta.\mathtt{ShareSize}(l) + \log(\mu) + \kappa + \alpha.\mathtt{ProofSize}(\mu) + \alpha.\mathtt{AccSize}).$$

Let us now bound the number of messages sent by honest parties S. For each share, the expected size of the neighborhood for a party is given by the probability of selecting each party as a neighbor times the number of parties. If we let $N \xleftarrow{\$} \mathsf{ERS}(\rho)$, then the expected size is given by $\mathrm{E}[|N|] = \rho \cdot n$. The total number of messages sent is upper bounded by the scenario where all parties are honest and forward a message for each share exactly once. By linearity of expectation, we get that the expected number of messages is given by

$$\mathrm{E}[S] \leq \mu \cdot \rho \cdot n^2.$$

By Chernoff (Lemma 1), we have that for any $\delta \in [0, 1]$,

$$\Pr\left[S \geq (1+\delta) \cdot \mu \cdot \rho \cdot n^2\right] \leq e^{-\frac{\delta^2 \cdot \mu \cdot \rho \cdot n^2}{3}}.$$

Hence, if $\mu \geq \kappa$ and $\rho = \Omega(n^{-2})$, then the probability that $S = O(\mu \cdot \rho \cdot n^2)$ is overwhelming in κ. Consequently, we have that the communication complexity is upper bounded by

$$O(\mu \cdot \rho \cdot n^2 \cdot (\zeta.\mathtt{ShareSize}(l) + \log(\mu) + \kappa + \alpha.\mathtt{ProofSize}(\mu) + \alpha.\mathtt{AccSize})), \tag{13}$$

with a probability that is overwhelming in κ. We further note that if additionally $\rho = \Omega(n^{-1})$, then the number of parties a single party has to connect with in the protocol is also bounded by

$$O(\rho \cdot n \cdot \mu), \tag{14}$$

by Chernoff (Lemma 1), with a probability overwhelming in κ. Thereby, the communication for each party will be upper bounded by:

$$O(\mu \cdot \rho \cdot n \cdot (\zeta.\mathtt{ShareSize}(l) + \log(\mu) + \kappa + \alpha.\mathtt{ProofSize}(\mu) + \alpha.\mathtt{AccSize})), \tag{15}$$

Below we state a simple corollary of Corollary 1 and Theorem 4 that bounds the necessary parameters in order to instantiate $\mathsf{WeakFlood2Flood}$ securely with $\mathsf{ERFlood}$.

Corollary 2. *There exists $\xi \in (0, 1]$ s.t. for any $n \geq 11 \cdot \gamma^{-1}$ there is a $\rho = O(n^{-1})$ and $\Delta = O(\log(n) \cdot \Delta_{NET})$ s.t. if $\mu \geq \frac{\log(n)+\kappa}{\xi}$, $e \geq \mu \cdot \left(1 - \frac{\xi}{2}\right)$, ζ is a (μ, e)-ECCS, and α is a WSCAS, then the protocol $\mathsf{ECFlood}(\rho, \zeta, \alpha)$ is a strong Δ-flooding protocol.*

Using Reed-Solomon codes and since $e \geq \mu \cdot \left(1 - \frac{\xi}{2}\right)$, we have that Equation (1) implies that each share size is bounded by:

$$\zeta.\mathtt{ShareSize} = O\left(\frac{l}{\mu \cdot \xi}\right) = O\left(\frac{l}{\mu}\right).$$

Furthermore, using efficient accumulators (see Equation (2)) that have accumulator and proof sizes of $O(\kappa)$ bits, and setting $\mu = \frac{\log(n)+\kappa}{\xi}$, we obtain from Equation (13) that the total communication complexity is bounded by

$$O(n \cdot (l + (\log(n) + \kappa) \cdot (\log(\log(n)) + \kappa))).$$

Note that this is optimal if

$$l = \Omega((\log(n) + \kappa) \cdot (\log(\log(n)) + \kappa)).$$

Similarly, by Equation (14) each party has at most $O(\log(n) + \kappa)$ neighbors and by Equation (15) the per-party communication is bounded by $O(l + (\log(n) + \kappa) \cdot (\log(\log(n)) + \kappa))$.

5 Flooding in the Weighted Setting

So far we considered the setting where a certain fraction of parties may behave adversarially. However, blockchain systems are often based on the assumption that the adversary controls a fraction of the total weight of a resource. Leveraging ideas from [LMM+22], we provide a general transformation for a flooding protocol in the equal-weights setting to security in the weighted setting.

Model. Before describing the transformation itself, we briefly recap the model of public weighted parties. 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 $\widetilde{\gamma} \in (0, 1]$ fraction of the total weight. Formally, we assume that,

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

and that all parties have a non-zero positive weight i.e. $\forall p \in \mathcal{P}, W_p > 0$.

Transformation. The main idea of our transformation is to let each party *emulate* a number of parties in another flooding protocol. We use the same emulation function as [LMM+22] where each weighted party $p \in \mathcal{P}$ emulates $\lceil \alpha_p \cdot n \rceil$ non-weighted parties. For each party $p \in \mathcal{P}$, we define a set of parties that this party emulates as

$$\mathtt{E}(p) \coloneqq \{p_i \mid i \in \mathbb{N} \wedge i \leq \lceil \alpha_p \cdot n \rceil\}.$$

Note that because all parties have a non-zero weight, all parties emulate at least one party, i.e., for any party $p \in \mathcal{P}$ we have $\mathtt{E}(p) \neq \varnothing$. For convenience, we introduce notation for the set of emulated parties, $\mathcal{P}_{\mathtt{E}} = \bigcup_{p \in \mathcal{P}} \mathtt{E}(p)$, the total number of emulated parties $n_{\mathtt{E}} = |\mathcal{P}_{\mathtt{E}}|$, the set of emulated parties that are emulated by honest players $\mathcal{H}_{\mathtt{E}} = \bigcup_{p \in \mathcal{H}} \mathtt{E}(p)$ and the number of honestly emulated parties $h_{\mathtt{E}} = |\mathcal{H}_{\mathtt{E}}|$. Following [LMM+22], we note that

$$n_{\mathtt{E}} = \sum_{p \in \mathcal{P}} \lceil \alpha_p \cdot n \rceil \leq \sum_{p \in \mathcal{P}} \alpha_p \cdot n + 1 = 2 \cdot n, \tag{16}$$

and

$$h_{\mathtt{E}} = \sum_{p \in \mathcal{H}} \lceil \alpha_p \cdot n \rceil \geq \widetilde{\gamma} \cdot n. \tag{17}$$

When defining a strong flooding protocol in Section 2.2, we were not explicit about the set of parties a flooding protocol has to provide guarantees for, as all of our previous flooding protocols have simply worked for the same set of assumed parties p. Below, this will not be the case, as the flooding protocols we discuss will work for a different set of parties. Hence, we will make these sets explicit by using the phrase that "a protocol is a flooding protocol for a set of parties".

Protocol Flood2WeightedFlood(Π)

The protocol is parameterized by a protocol Π that is a flooding protocol for $\mathcal{P}_{\mathtt{E}}$.
Each party $p \in \mathcal{P}$ starts a process for each of their emulated parties, and lets these processes participate in the protocol Π.

Initialize: Initially, each party p initialize all of their emulated parties $\mathtt{E}(p)$ in Π.

Send: When p receives (Send, m) they pick $p_i \in \mathtt{E}(p)$ and forward (Send, m) to p_i in Π.

Get Messages: When p receives $(\mathit{GetMessages})$ they pick $p_i \in \mathtt{E}(p)$ forward $(\mathit{GetMessages})$ to p_i in Π, and return the set of messages returned to p_i.

Below we prove that if Flood2WeightedFlood is instantiated with a strong flooding protocol for $n_{\mathtt{E}}$, then Flood2WeightedFlood will itself be a strong flooding protocol.

Theorem 6. *Let $\Delta \in \mathbb{N}$. If Π is a strong Δ-flooding protocol for $\mathcal{P}_{\mathtt{E}}$ under the assumption that at least $\widetilde{\gamma} \cdot n$ of them behaves honestly, then Flood2WeightedFlood(Π) is a strong Δ-flooding protocol for $\mathcal{P}$.*

Proof. Let m be a message that is input to some honest party at time t. Since all parties emulate at least one party, this implies that the message will also be input to some honest emulated party in Π at time t. Because Π is a strong Δ-flooding protocol for $\mathcal{P}_{\mathtt{E}}$ when $\widetilde{\gamma} \cdot n$ parties are honest (Equation (17) ensures that this is actually the case), then there is an overwhelming probability in κ that all emulated parties receive m before $t + \Delta$. As each honest party emulates at least one party, this implies that all honest parties will also receive the message with a probability that is overwhelming in Π. □

Realising a strong flooding protocol for $\mathcal{P}_{\mathtt{E}}$. At first glance, it may seem like that Theorem 6 allows us to translate the protocols presented in Section 4 to the weighted setting without further work. However, even though the protocols in these sections work for any set of parties, they make channels, which are only assumed for the actual set of parties $\mathcal{P}$ and not for the emulated set of parties $\mathcal{P}_{\mathtt{E}}$. To use these protocols blackbox in the weighted setting, we, therefore, need to show how to establish channels between the emulated parties.

We note that channels for the emulated set of parties can easily be established from channels between the original set of parties. One way to do this is by simply prepending $(p_e, p_{e'})$ to any message that an emulated party p_e wishes to send to another emulated party p'. When a party receives such a message on a normal channel, they will take it as an input on the emulated channel between the emulated parties p_e and $p_{e'}$.

Communication complexity analysis. We note that the complexity analysis in Section 4.3 also applies when the protocol is transformed to work for the weighted setting because $n_{\mathtt{E}} = O(n)$ (Equation (16)) and the fraction of honest emulated parties $\frac{h_{\mathtt{E}}}{n_{\mathtt{E}}} = O(\widetilde{\gamma})$ (by Equations (16) and (17)). The only thing that changes is that all messages will have identifiers for emulated parties prepended. The size of such identifiers is bounded by $O(\log(n))$. When this is threaded

through the analysis using the same parameters as in Section 4.3, we see that for a suitable ρ, ζ and α, the communication complexity of the Flood2WeightedFlood(ECFlood(ρ, ζ, α)) is bounded by

$$O(n \cdot (l + (\log(n) + \kappa)^2)).$$

That is Flood2WeightedFlood(ECFlood(ρ, ζ, α)) has an optimal communication complexity when $l = \Omega((\log(n) + \kappa)^2)$. It is, however, worth noting that for our particular protocol, it is not necessary to keep the messages delivered to different emulated parties separate. In particular, Flood2WeightedFlood(ECFlood(ρ, ζ, α)) would have the same guarantees if any message sent from an emulated party of party p_i to an emulated party of p_j is simply delivered to *all* emulated parties of p_j. In that case, the communication complexity of Flood2WeightedFlood(ECFlood(ρ, ζ, α)) would be optimal under the same constraints as ECFlood(ρ, ζ, α).

6 Estimating Practical Parameters for ECFlood Using Probabilistic Simulations

In Section 4.3, it was shown that parameters could be set such that ECFlood theoretically constitutes an asymptotically optimal flooding algorithm. In order to show this, we instantiated many variables to constants required by our analysis, but these are most likely not instantiated anywhere near optimally, and nor are our analyses themselves optimal. To provide a guideline for how to instantiate the parameters of our protocol for practical performance, we made probabilistic simulations of the protocol executions that explore how the message complexity of our protocols is affected by adjusting parameters.

6.1 Setup for Simulations

We implemented a probabilistic experiment among n nodes where $\frac{n}{2}$ of them behave honestly and will actually forward any received message according to our protocol if they receive a message. We consider not forwarding any messages as the worst-case behavior of an adversary. Therefore, the remaining $\frac{n}{2}$ parties, which we consider corrupt, will not participate in forwarding any messages sent to them. That is, an initial party will distribute several messages by selecting a set of neighbors using ERS$(\frac{d}{n})$ (for varying values of d) for each message. These will then again forward each message to a random set of neighbors. This continues until no honest party receives the message for the first time. For simulations of ERFlood, a single message is initially input, whereas, for ECFlood, several messages will be input corresponding to the number of shares a message is divided into in the protocol.

We have repeated our simulations 1000 times for each set of parameters. Below, we report on various statistics from these simulations.

6.2 Estimating Parameters for ERFlood

In Section 4.3 it was shown that the communication complexity of WeakFlood2Flood instantiated with ERFlood$(\frac{d}{n})$ is directly proportional to $\frac{d}{\xi}$, if ERFlood constitutes a weak (Δ, ξ) flooding protocol for some Δ and ξ.

Initially, let us ignore how Δ is affected by changing the expected degree d and let us focus solely on finding estimates of $\frac{d}{\xi}$ for different expected degrees. A simple way to estimate the maximum ξ for which ERFlood$(\frac{d}{n})$ is a ξ-weak flooding algorithm is to select a party different from the sender and count the rate with which this party receives a message throughout many executions. The results of this approach for different values of n and d can be found in Figure 3.

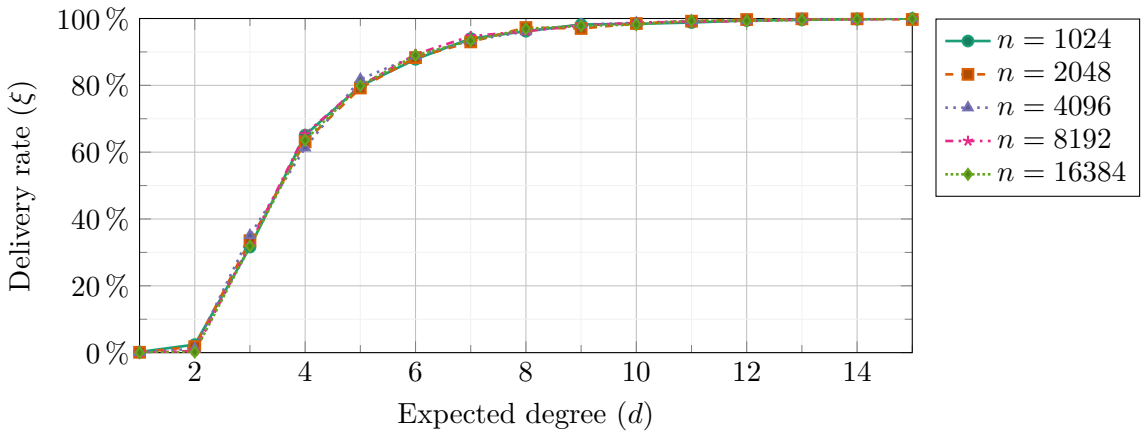

Figure 3: Results for simulations of $\mathsf{ERFlood}(\frac{d}{n})$ for different values of d and n. The delivery rate for a fixed party different from the sender is plotted as a function of changing the expected degree d.

It is noteworthy that the graphs for different values of n are extremely close to each other. When $\mathsf{ERFlood}(\frac{d}{n})$ is executed, it is clear that d will be the expected size of the neighborhood of each party. Hence, our simulations confirm that the delivery rate of $\mathsf{ERFlood}(\frac{d}{n})$ (and thereby how well it acts as a weak flooding algorithm) is truly independent of the number of parties in the protocol n. In particular, this holds even for very small values of d (and n).

It is striking that even for $d = 3$, where an honest party in expectation forwards the message to just 1.5 other honest parties, the probability that a particular party receives the message seems to be about $\sim 30\%$. To understand this behavior, we collected additional statistics on the fraction of parties reached in each execution of the protocol. This data is presented in Figure 4.

Interestingly, it seems that for all the different degrees plotted; there is an initial drop before the curves reach a plateau, where they stay for a while. Our interpretation of this phenomenon is that there is quite a high probability that the initial sender or their close neighbors talk only to dishonest parties, which stops the propagation. However, once a critical mass of parties has been reached, it becomes very unlikely that they all select only dishonest parties or parties that have already received the message as their neighbors (which is what is required to make the propagation of the message stop). So even for $d = 3$, where on average each party only talks to 1.5 honest parties, there is more than 50% chance that the message will spread to more than 50% of the parties.

With estimates on ξ, we are also able to estimate the size of the factor $\frac{d}{\xi}$ that is multiplied by the message length in the communication complexity. In Figure 5, we plot this. Our simulations indicates that the best value for d in order to get the lowest overall communication complexity is ~ 5 where $\frac{d}{\xi}$ will be just above 6.

We now turn our attention to how the Δ parameter of the weak flooding algorithm $\mathsf{ERFlood}(\frac{d}{\xi})$ is affected by varying the value d. To provide an upper bound on Δ, we recorded the maximum number of hops from the sender to any other party that receives the message in each execution. Note that the number of hops times the maximum delay on the point-to-point channels directly translates to an upper bound on the delivery time for a message in a real execution of the protocol. In Figure 6, we plot the maximum number of hops across all the simulations (for a single set of parameters) for the various number of parties n and expected degrees d. We include only those degrees that are of interest w.r.t. an overall low communication complexity.

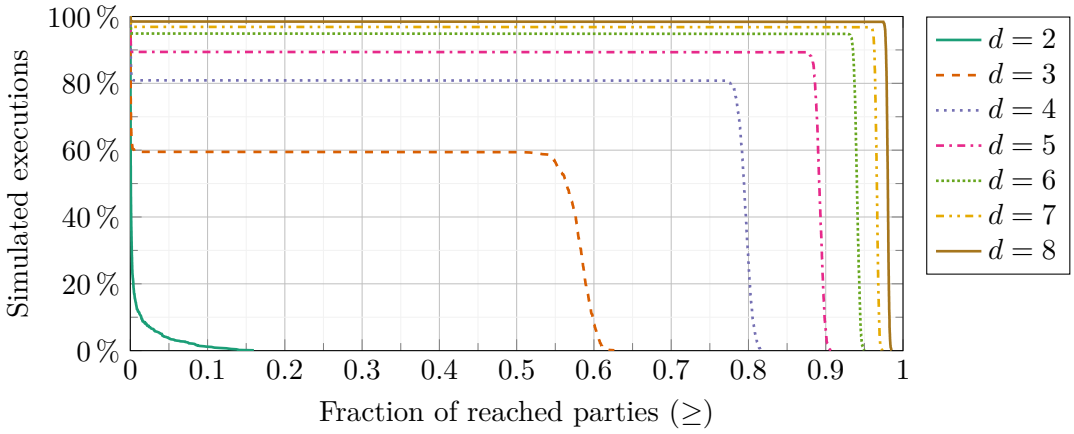

Figure 4: Results for simulations of $\mathsf{ERFlood}(\frac{d}{n})$ for different values of d and a fixed $n = 8192$. The graphs show the percentages of the simulated executions (on the y-axis) where at least a certain fraction of parties received the message (on the x-axis).

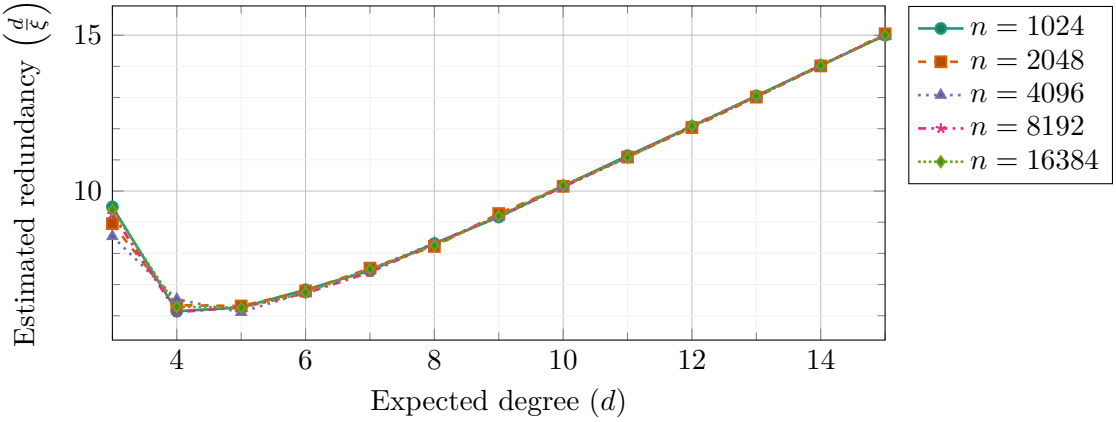

Figure 5: Results for simulations of $\mathsf{ERFlood}(\frac{d}{n})$ for different values of d and n. The graphs show the expected degree normalized by the observed delivery rate as a function of the expected degree.

We note that during the execution, no message was delivered in more than 33 hops for expected degrees at least 4. For degrees at least 5, this maximum number of observed hops drops to 23, and for degrees larger than 10, no message is delivered in more than 10 hops.

6.3 Estimating Parameters for ECFlood

Even though we in the previous section established that $\mathsf{ERFlood}(\frac{d}{n})$ delivers the "best" weak flooding network when d is approximately 5, this does not necessarily translate to the protocol WeakFlood2Flood instantiated with $\mathsf{ERFlood}(\frac{d}{n})$. The reason is that WeakFlood2Flood also needs to be instantiated with an (μ, e)-ECCS, and the redundancy of the ECCS scheme is proportional to $(\mu - e)^{-1}$. For WeakFlood2Flood to be secure, we need that all parties receive at least $\mu - e$ shares. In expectation, all parties will receive $\xi \cdot \mu$ when instantiated with a weak ξ-flooding protocol. However, for a secure combined protocol, we need that the probability that some parties receive significantly less than $\xi \cdot \mu$ shares is small.

In this section, we provide guidelines for how to select d, μ, and e for a ζ that is (μ, e)-ECCS,

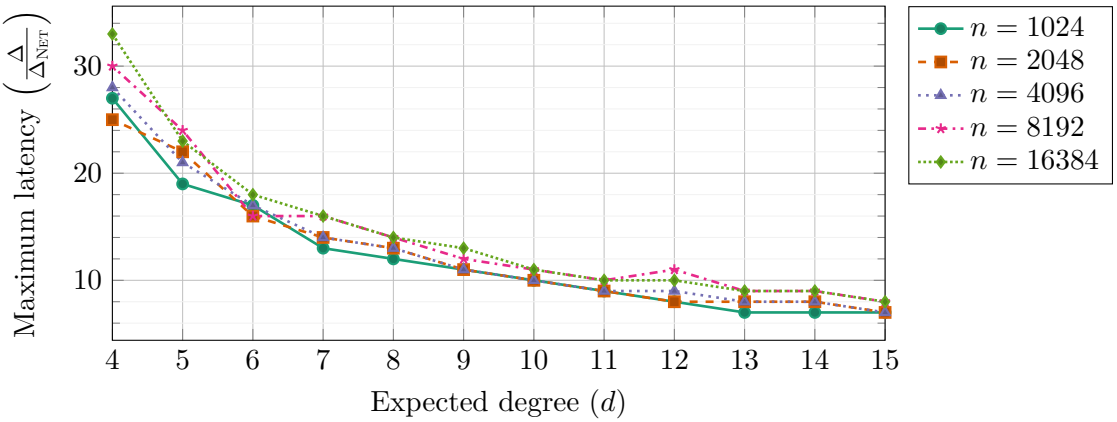

Figure 6: Results for simulations of $\mathsf{ERFlood}(\frac{d}{n})$ for different values of d and n. The graphs show the maximal latency from the sender to any party that receives the message for different values of d.

such that $\mathsf{ECFlood}(d \cdot n^{-1}, \zeta)$ is both secure and has a small overall factor multiplied to $n \cdot l$ in the communication complexity. We will abuse notation slightly and ignore the WSCAS scheme that is also a parameter of $\mathsf{ECFlood}$, as we will not do any simulations w.r.t. the efficiency of such scheme. Sometimes we will also leave out the ECCS of the notation and treat the number of shares explicitly.

To estimate how to set e of the ECCS, we record the minimum fraction of shares received by any party in any of the simulations (using the same parameters), and plot how this minimum fraction of received shares across all simulations, β, is affected by a varying number of shares μ and degrees d. The results of this can be found in Figure 7.

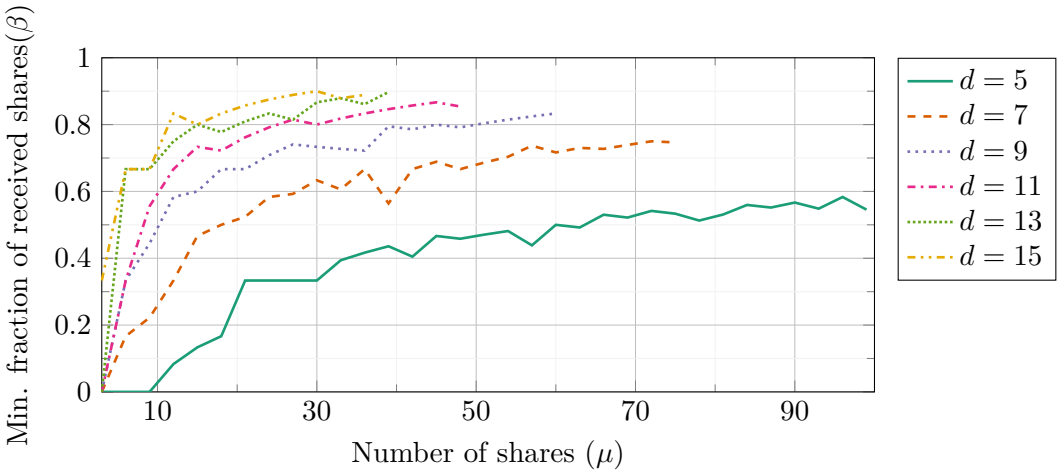

Figure 7: Results for simulations of $\mathsf{ECFlood}(\frac{d}{n})$ for different values of d, a fixed number of parties $n = 8192$, and a variable number of shares μ. The graphs show the minimum fraction of shares received by any party in the simulations for different number of expected neighbors $\mu \cdot d$. The number of shares is incremented in steps of 3.

Note that if one had instantiated e such that $\mu - e = \beta \cdot \mu \iff e = \mu \cdot (1 - \beta)$ then all parties would have been able to reconstruct the message in all simulations. For these parameters, the protocol would have a total communication complexity of roughly $\beta^{-1} \cdot d \cdot n \cdot l$. Therefore, we

will refer to $\beta^{-1} \cdot d$ as the redundancy of the protocol.

In Figure 1, we plot the redundancy when letting $e = \mu \cdot (1 - \beta)$ as a function of the expected size of the neighborhood $\mu \cdot d$ for a varying number of shares μ and degrees d. We do not plot the redundancy for $d = 3$ and $d = 4$ as it for all considered values of μ will be so high that it is of no interest. Note that even though the expected redundancy has a minimum around $d = 5$ (according to Figure 5), it seems that the actual redundancy of protocols instantiated securely for relatively small expected neighborhoods ≤ 100 is as small for degree $d = 7$. This is because if ξ is higher (provided by a higher degree), then it will concentrate more quickly around the mean (when μ increases) than when ξ is relatively small. Suppose you are willing to accept having 120 outgoing connections. In that case, $d = 7$ and $\mu = 20$ will deliver a flooding protocol that in non of the simulations fails and has a redundancy of just 15. If you are willing to have larger neighborhoods, then we can instantiate our protocol such that redundancy goes below 10 (for example, with $d = 5$ and $\mu = 70$).

We note that in terms of redundancy, this is a significant improvement over the protocol presented in [LMM+22]. Simulations in [LMM+22, Figure 3] showed that in order to get a flooding protocol that succeeds 99% of the time for 8192 parties, one would, in their protocol, have to let each party forward the entire message to around 28 neighbors. This will also be their worst-case redundancy, and hence the protocol presented in this work requires less than half the communication complexity for sufficiently long messages. We conclude that ECFlood is not only asymptotically optimal but seem to have advantages over existing byzantine fault-tolerant flooding protocols within the practical range of parameters.

For a specific number of shares, $\mu = 20$, we additionally record the percentages of shares where all parties received at least a certain fraction of shares for different parameters d. The results are plotted in Figure 8. The maximum fraction of shares that all parties in all simulations received in Figure 8 corresponds to β in Figure 7. From the plot, it can be seen that if one is willing to accept a low rate of failing executions where not all parties can reconstruct, then one can instantiate e slightly lower than $\mu \cdot (1 - \beta)$.

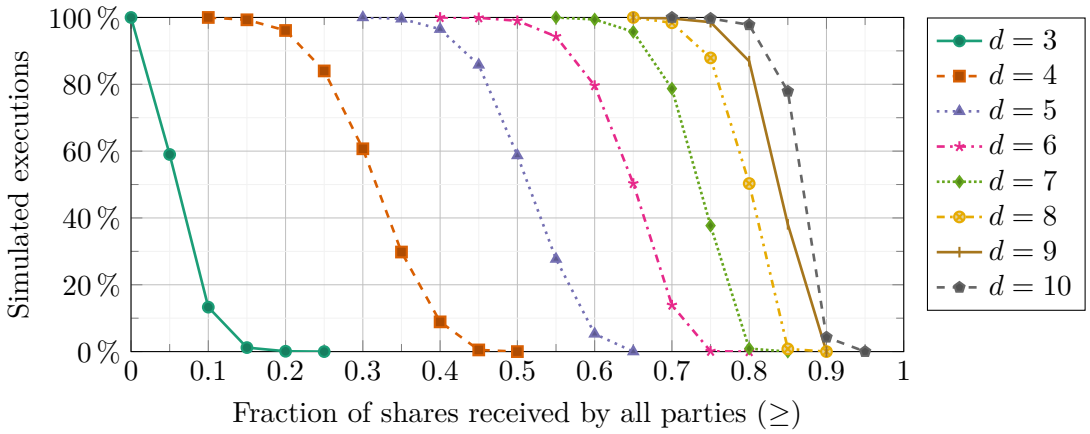

Figure 8: Results for simulations of ECFlood($\frac{d}{n}$) for different values of d, a fixed number of parties $n = 8192$, and a fixed number of shares $\mu = 20$. The graphs show the percentages of the simulated executions (on the y-axis) where all parties received at least a certain percentage of all the shares sent out by the sender.

Finally, we count the maximum number of hops throughout the executions for any party to have received the minimum fraction of shares (at which point all parties could reconstruct the message if parameters were set accordingly). We plot this for a fixed number of shares

$\mu = 30$ and a varying number of parties μ and internal parameter d in Figure 2. The reason that this is plotted for a fixed number of shares is that we observed that it does not change when increasing the shares. Therefore the results are representative of latencies for all numbers of shares discussed previously.

Surprisingly, the latency for ECFlood is significantly lower than the latency of ERFlood (Figure 6). We believe this is because it is a rare event that the maximum latency of ERFlood will occur. Therefore it will become very unlikely that such rare events coincide for shares sent to the same party that receives the minimum number of shares. We note that by adjusting the parameter d, ECFlood can be tuned to achieve a similar latency to that of the protocol from [LMM+22] while still maintaining a significantly lower redundancy.

As noted in Section 3, the protocol ECCast, has a redundancy of only γ^{-1} (which for the parameters of our simulations are only 2) and a latency of just 2. However, to achieve this low redundancy and latency, ECCast requires each party to talk to 8191 neighbors, and the two protocols, therefore, present a tradeoff between low redundancy and diameter and a low number of neighbors.

7 Conclusion

We presented two protocols for message dissemination, ECCast and ECFlood, that are asymptotically optimal w.r.t. communication complexity and per-party communication for sufficiently long messages. Our simulations of ECFlood indicate that the protocol is not only asymptotically optimal but actually offers significant improvements in terms of redundancy over existing provably secure protocols for parameters within the practical realm. The two protocols present a tradeoff between a low number of neighbors of each party and the communication complexity and latency on the other side. Additionally, we presented the protocol Flood2WeightedFlood, which allows our protocols to be used in the weighted setting, which is typical for blockchains.

For blockchain protocols where parties' bandwidth is a limiting factor for the protocol's throughput, our results allow to increase the throughput of the protocol, e.g., by increasing the size of blocks or by simply benefiting from decreased latency of sending fewer bits on the underlying point-to-point channels.

References

[BdM93] Josh Cohen Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In *EUROCRYPT*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285. Springer, 1993.

[BLZLN22] Amey Bhangale, Chen-Da Liu-Zhang, Julian Loss, and Kartik Nayak. Efficient adaptively-secure byzantine agreement for long messages. *ASIACRYPT*, 2022.

[BP97] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 480–494, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[Can20] Ran Canetti. Universally composable security. *J. ACM*, 67(5):28:1–28:94, 2020.

[CKMR22] Sandro Coretti, Aggelos Kiayias, Cristopher Moore, and Alexander Russell. The generals' scuttlebutt: Byzantine-resilient gossip protocols. In *CCS*, pages 595–608. ACM, 2022.

[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.

[Did09] Frédéric Didier. Efficient erasure decoding of reed-solomon codes. *CoRR*, abs/0901.1886, 2009.

[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.

[FH06] Matthias Fitzi and Martin Hirt. Optimally efficient multi-valued byzantine agreement. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 163–168, 2006.

[FOA16] Muntadher Fadhil, Gareth Owenson, and Mo Adda. A bitcoin model for evaluation of clustering to improve propagation delay in bitcoin network. In *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, pages 468–475, 2016.

[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.

[GP16] Chaya Ganesh and Arpita Patra. Broadcast extensions with optimal communication and round complexity. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 371–380, 2016.

[GRKC15] Arthur Gervais, Hubert Ritzdorf, Ghassan O. Karame, and Srdjan Capkun. Tampering with the delivery of blocks and transactions in bitcoin. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 692–705, New York, NY, USA, 2015. Association for Computing Machinery.

[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.

[KYA22] Ioannis Kaklamanis, Lei Yang, and Mohammad Alizadeh. Poster: Coded broadcast for scalable leader-based BFT consensus. In *CCS*, pages 3375–3377. ACM, 2022.

[LMM+22] Chen-Da Liu-Zhang, Christian Matt, Ueli Maurer, Guilherme Rito, and Søren Eller Thomsen. Practical provably secure flooding for blockchains, 2022.

[Mer89] Ralph C. Merkle. A certified digital signature. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, 1989.

[MNT22] Christian Matt, Jesper Buus Nielsen, and Søren Eller Thomsen. Formalizing delayed adaptive corruptions and the security of flooding networks. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 400–430, Cham, 2022. Springer Nature Switzerland.

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

[Ngu05] Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2005.

[NRS+20] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. Improved extension protocols for byzantine broadcast and agreement. In *DISC*, 2020.

[ÖMBS21] Ilker Özçelik, Sai Medury, Justin T. Broaddus, and Anthony Skjellum. An overview of cryptographic accumulators. In *ICISSP*, pages 661–669. SCITEPRESS, 2021.

[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.

[RS60] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of The Society for Industrial and Applied Mathematics*, 8:300–304, 1960.

[RT19] Elias Rohrer and Florian Tschorsch. Kadcast: A structured approach to broadcast in blockchain networks. In *AFT*, pages 199–213. ACM, 2019.

[TC84] Russell Turpin and Brian A. Coan. Extending binary byzantine agreement to multivalued byzantine agreement. *Inf. Process. Lett.*, 18(2):73–76, 1984.

[VT19] Huy Vu and Hitesh Tewari. An efficient peer-to-peer bitcoin protocol with probabilistic flooding. In Mahdi H. Miraz, Peter S. Excell, Andrew Ware, Safeeullah Soomro, and Maaruf Ali, editors, *Emerging Technologies in Computing*, pages 29–45, Cham, 2019. Springer International Publishing.

[W+14] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[YMR+19] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *PODC*, pages 347–356. ACM, 2019.