Sub-Linear Overhead in Static Schedules for Fault-Tolerant Transmission

Zhe Wang
Washington University in St. Louis
zhe.wang@wustl.edu

Kunal Agrawal
Washington University in St. Louis
kunal@wustl.edu

Jeremy T. Fineman
Georgetown University
jf474@georgetown.edu

2021 IEEE Real-Time Systems Symposium (RTSS) | 978-1-6654-2802-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/RTSS52674.2021.00044

***Abstract*—Shared communication media are widely used in many applications including safety critical applications such as control systems on flights or autonomous vehicles. Noise and transient errors can cause transmission failures. We consider the problem of designing fault tolerant static schedules for transmitting messages in these media. In particular, we assume that the schedule of transmission over all messages must be computed in advance and must guarantee that all messages will be delivered as long as the number of medium errors falls below a provided upper bound, regardless of when the medium errors occur. It is crucial that the messages be delivered in a timely manner, and hence we are interested in minimizing the length of the schedule that achieves the desired level of fault tolerance. In this paper, we provide an efficient algorithm for producing a schedule for n messages with total length $n + O(f^2 \log^2 n)$ that can tolerate f medium errors. We also prove that fault-tolerant schedules with length $n+O(f \log f \log n)$ exist. Since n steps are required to transmit n messages, the overhead of fault tolerance is characterized by the additive terms of $O(f^2 \log^2 n)$ and $O(f \log f \log n)$, respectively. Both of these terms are sublinear in n and represent asymptotic improvements to the previously best known schedule, which has overhead $fn/2$.**

***Index Terms*—Fault-tolerant transmission; Static scheduling**

I. Introduction

Shared communication medium such as a shared bus, wireless network, CAN, are commonly used in myriad systems. On safety-critical systems and/or on embedded devices, it is often beneficial to generate a ***static schedule*** — where time for sending each message is predetermined — since these enable both simpler implementation and a deterministic upper bound on message delivery time. For instance, static schedules are used on time-triggered systems, like TTA [1], where timeline is divided into a series of identical and exclusive time slots, and messages are sent in pre-allocated slots.

A transmission algorithm on (especially wireless) channels for messages must tolerate ***noise*** or transmission errors in the medium — such noise leads to message failure or corrupts the message. When a message encounters noise, it must be sent repeatedly until it is correctly transmitted. Therefore, a static schedule for message transmission must reserve sufficient slots for each message to tolerate medium errors in transmission.

Problem Statement and Prior Work In this paper, we consider the scenario where we have n messages to be transmitted (possibly by different senders) on a shared medium where time is divided into slots. We assume that the channel has upto f medium errors — that is, upto f time slots experience noise. In addition, if more than one message is transmitted in the same time slot, a ***collision*** occurs and neither message will be sent. If a time slot experiences a medium error or a collision, the message(s) transmitted during the slot fail. We assume that the sender knows when a message fails and must retransmit. [1] In static schedules, each sender has a list of slots. It transmits its message on each successive slot assigned to it until the message succeeds and then stops transmitting the message. We assume that senders are not aware of and cannot react to other runtime conditions (such as which other messages have successfully transmitted or how many medium errors or collisions have occurred so far).

The goal is to generate a ***fault-tolerant static schedule*** which minimizes ***makespan*** while guaranteeing that all n messages are transmitted successfully for all possible settings of up to f slots experiencing medium errors. This strong correctness requirement means that we do not make any assumptions about the distribution of medium errors and the schedule must guarantee that all messages successfully transmit even if an adversary that knows the static schedule generates (upto f) medium errors. We are interested in finding fault-tolerant schedules with length as short as possible. Even with $f = 0$, all schedules must have length at least n (and in fact with $f = 0$ achieving length n is trivial). We thus focus on how much we need to add to the schedule length in order to achieve fault tolerance as f increases. Specifically, if the schedule has length $n + T$, we say that the ***fault-tolerance overhead***, or simply the ***overhead***, of this schedule is T. The goal is to minimize this overhead.

It is clear that each message must be assigned to at least $f + 1$ slots under this adversarial model — if any message m is assigned to fewer slots, the f medium errors can occur on all the slots where m is assigned and m will fail to transmit. The simplest schedule is to send each message for $f+1$ times which uses $n + n \times f$ slots and the overhead is $n \times f$. This is the best we can do if we assign at most one message to each slot in the static schedule. One can take advantage of the fact that the sender knows when a message has transmitted successfully and will not transmit such a message again to generate shorter schedules. The state-of-the-art schedule [3] uses $n + n \times f/2$ slots and the overhead is $n \times f/2$, which

[1] This is a fairly standard assumption and is often implemented using some sort of acknowledgment mechanism, like the use of a short "acknowledgement" frame in AirTight [2].

978-1-6654-2802-6/21/$31.00 ©2021 IEEE
DOI 10.1109/RTSS52674.2021.00044

is better than the naive one. These overheads increase linearly along with n — as n increases, the overhead can be quite large even for a small f.

Main Results Can the fault-tolerant overhead increases sublinearly in the number of messages n? In this paper, we present techniques for designing static schedules with sublinear overheads for the model described above. In particular, we provide two mappings, one with overhead $O(f^2 \log^2 n)$ and the other with overhead $O(f \log f \log n)$. While the problem (fixed number of messages and medium errors) and the objective (minimize overhead) considered may not directly apply to real systems, this simple model allows us to investigate whether sublinear overheads are even possible using static schedules and we answer this question in the affirmative. We believe that these insights can be applied to more complex and realistic problems as well.

Contributions The main results of this paper are:

- (Section III.) We give an algorithm for generating fault-tolerant schedules with overhead $O(f^2 \log^2 n)$ and hence total length $n+O(f^2 \log^2 n)$. The overhead depends only logarithmically on n and is asymptotically better than the state of the art algorithm [3] whenever $f = o(n/\log^2 n)$, and can be significantly better when f is much smaller that n. This mapping can be generated efficiently.
- (Section IV.) We prove the existence of fault-tolerant schedules with overhead $O(f \log f \log n)$ and hence the static schedule length is $n + O(f \log f \log n)$. The algorithm is randomized, and generates a fault-tolerant mapping with a non-zero probability. Although the algorithm for generating the schedule is efficient, verifying that the schedule is fault tolerant takes time exponential in f. Therefore, this may not be a viable approach for large f, but can be practical for small f since the schedule need be generated just once offline.
- (Section V.) While our methods are asymptotically better than prior methods, they do have constant factors. We analytically compute schedule lengths of our methods and compare against the method proposed by Agrawal et al. [3] for concrete values of n and f to verify that under certain conditions, sublinear mappings are shorter.

The main technical building block for our schedules is something called an α-good mapping. (We discuss these mappings in Section II.) Bender et al. [4] introduced α-good mappings for a different problem that also uses some degree of static scheduling. In their setting, the main issue is coping with collisions, not fault tolerance. It turns out that we can leverage the same α-good mappings to achieve fault tolerance. Doing so involves either proving stronger properties about the mapping (Section III), or carefully integrating multiple mappings of different sizes (Section IV).

II. Background

This section provides definitions as well as relevant background on α-good mappings [4].

We build static schedules by concatenating multiple subschedules that we call ***mappings***. A mapping is defined over a contiguous interval of slots, and it specifies which messages transmit in each slot. There is no technical difference between the terms, but we use the word mapping to refer to components of the schedule and schedule to refer to the whole. We assume throughout that messages have distinct IDs from $1, 2, \ldots, n$ and use these to define mappings. The simplest mapping is the identity mapping.

Definition 1 (Identity mapping — $I(n)$)**.** *The identity mapping for n messages spans n slots. Message i is mapped to slot i.*

There is no apparent advantage in assigning multiple messages to a slot the first time a message is scheduled. All fault-tolerant schedules thus begin with the identity mapping of length n. We refer to what follows the identity mapping as the ***retransmission mapping*** or ***retransmission schedule***. The retransmission mapping is the interesting part of fault-tolerant schedules and the overhead due to fault-tolerance is the length of the retransmission mapping. For example, the dual mapping [3] has retransmission mapping length of $nf/2$.

We will rely on the notion of α-good mappings introduced by Bender et al. [4] to generate sublinear length retransmission mappings. It is a property that (lower) bounds the number of collision-free slots with respect to subsets of remaining messages. Here $X = \{1, 2, \ldots, n\}$ is the set of n message IDs and $m \leq n$ is the size of the subsets. As given by the mathematical definition below, a mapping is said to be ***α-good with respect to subset size m*** if: for all subsets $S \subset X$ of size m, at least an α fraction of the messages in S are mapped to a slot that does not contain any other messages from S. This means that if we have m remaining messages to transmit, at least αm messages are, effectively, collision free, no matter which m messages remain.

Definition 2 (α-good mapping)**.** *Let X be the set of message IDs and let $n = |X|$. Consider a mapping and let $slots(x)$ denote the set of slots to which $x \in X$ is assigned. For a given $0 \leq \alpha \leq 1$ and a given subset size $m \leq n$, a mapping is an **α-good for subset size m** if the mapping satisfies: $\forall S \subseteq X$ with $|S| = m$, $\exists Z \subseteq S$ with $|Z| \geq \alpha m$ such that $\forall x \in Z$, $\exists s \in slots(x)$ with $s \notin \bigcup_{y \in S \setminus x} slots(y)$.*

To clarify this concept, consider some examples. The first example of an α-good mapping with $\alpha = 1$ and $n = 4$ and $m = 2$:

$$\left\langle \{X_1, X_2\}\{X_2, X_3\}\{X_3, X_4\}\{X_1, X_4\} \right\rangle ,$$

where X_i represents the message i and "$\{\}$" represents a slot — this schedule contains 4 slots, and each slot contains 2 messages. Since $m = 2$, the mapping must ensure that for each pair of messages x, y, there exists a slot that contains x and not y and vice versa. To see why this is useful, consider that X_3 and X_4 have transmitted before this schedule, but X_1 and X_2 remain. If there are no errors during the transmission of this schedule, then X_2 will successfully transmit during slot 2 and X_1 will successfully transmit during slot 4. This is true

regardless of which two messages remain at the beginning of the mapping. Note that this mapping is not particularly short — we could also use the schedule where each message transmits in its own slot, which is trivially 1-good; however, for larger values of n and m, non-trivial 1-good schedules where messages share slots can be shorter, as we will soon see.

Now let us consider an example where $\alpha = 1/2$. We have $n = 4$ messages X_1, X_2, X_3, X_4 and $m = 2$. It turns out that the schedule

$$\langle \{X_1, X_2\}\{X_2, X_3\} \rangle$$

is $1/2$-good. Note that for a $1/2$-good mapping, we have fewer constraints. Now for each pair of messages x, y, **either** x has to have a slot which does not contain y or y has to have a slot that does not contain x. This allows us to generate this counter-intuitive mapping where X_4 is never transmitted. For instance, consider X_1 and X_4 — since the first slot contains X_1, but not X_4, it satisfies the condition. We can check similarly for all pairs.

The advantage of α-good mappings is realized when the size of the mapping is relatively small. Bender et al. [4] give the following methods for creating a 1-good mapping and a $1/2$-good mapping with sizes $O(m^2 \log^2 n)$ and $O(m \log n)$, respectively. Importantly, for both methods if $m \ll n$, then the number of slots is far below n. Both of their constructions involve concatenating several simple mappings that they call ***collections***. A collection is a mapping in which each message is assigned to exactly one slot.

We first describe the ***modulo mapping*** which is a small 1-good mapping.

Definition 3 (Modulo mapping)**.** *Let n be the number of messages and let $1 \leq m \leq n$ be a parameter of the mapping. Let $C = m\lceil \log n / \log(m \log n) \rceil$, and let $p_1, p_2, \ldots, p_C$ denote the C smallest prime numbers greater than $m \log n$. The modulo mapping consists of the concatenation of C submappings called collections. Collection i spans p_i slots numbered consecutively from 0, and all messages $x \in X$ with $x \equiv j \pmod{p_i}$ are assigned to slot j in collection i.*

Bender et al. [4] prove the following bound on the length of this mapping — it follows from the density of primes, allowing all primes considered to have value $O(m \log n)$.

Lemma 1 (From [4])**.** *The modulo mapping with parameter m is a 1-good mapping for subset size m, and it spans a total of $O(m^2 \log^2 n)$ slots.*

Without doing the full proof, here is the intuition for why the modulo mapping is 1-good. Consider two integers j and k where $j \neq k$. In the modulo mapping, j and k collide in collections i iff $j \equiv k \pmod{p_i}$, where p_i is the prime used for collection i. Consider the first $C_0 = \log n / \log(m \log n)$ collections. For j and k to collide in all these collections, we must have $|j - k| \geq \prod_{i=1}^{C_0} p_i$ for all primes p_i used in these collections. Since all the primes p_i are large ($\geq m \log n$), this necessarily implies $|j - k| \geq \prod_{i=1}^{C_0} p_i \geq (m \log n)^{C_0} = n$. Therefore, if our messages are numbered from 1 to n, the difference between the two numbers is at most $n - 1$, and no two messages can collide in all C_0 collections. This observation can be generalized: In particular, since we have a total of $C = mC_0 = m \log n / \log(m \log n)$ collections, we can show that there exists a collection such that j will not collide with any other subset of $m - 1$ messages implying that this is a 1-good mapping. Now consider the size of the mapping. The size of the mapping is the $\sum_{i=1}^{C} p_i$ where p_i is the ith prime number larger than $m \log n$. Since primes are relatively dense, any interval of size $N \log N$ contains approximately N primes for sufficiently large N. Therefore, if we consider an interval $(m \log n, km \log n)$ for some constant k, we will find C primes. Thus, $p_i = O(m \log n)$ for all p_i, implying that the mapping size is $O(m^2 \log^2 n)$.

Definition 4 (Ball-bin method)**.** *Let n be the number of message IDs and let $1 \leq m \leq n$ be a parameter of the mapping. The mapping is also parameterized by constant $d, c > 0$. The ball-bin mapping is the concatenation of $\lceil d \log n \rceil$ collections, each spanning $\lceil cm \rceil$ slots. In each collection, each message is assigned to one slot independently and uniformly at random.*

By construction, the number of slots used by the ball-bin method is bounded. But because ball-bin method is randomized, the resulting mapping is not always α-good. Bender et al. [4] show that the ball-bin method is likely to produce a $1/2$-good mapping, as stated by the following lemma. Note that the method simply returns a mapping, without any indication of whether the mappings is $1/2$-good or not. Thus, we need to verify the quality of the mapping before incorporating it into the fault-tolerant schedule, repeating the random generation in the unlucky event that the mapping is not $1/2$-good.

Lemma 2 (From [4])**.** *There exist constant settings of parameters $c > 0$, $d > 0$ and constant probability $p > 0$ such that: for any setting of parameter m, with probability at least p the ball-bin method produces a mapping that is $1/2$-good for subset size m. Moreover, the mapping produced always uses $O(m \log n)$ slots.*

III. Generating fault-tolerant mapping with divisible 1-good mappings

This section uses 1-good mappings to produce a fault-tolerant schedules. As mentioned in Section II, a fault-tolerant schedule consisting of an identity mapping followed by a retransmission mapping must ensure that all messages are transmitted even if there are up to f errors during the transmission regardless of which slots have the errors. We will now design retransmission mappings using 1-good mappings. We first argue that a single arbitrary 1-good mapping as the retransmission mapping does not guarantee fault-tolerance. We then define an additional condition on 1-good mappings, called ***divisibility***, and argue that the identity mapping followed by a divisible 1-good mapping is sufficient to guarantee fault tolerance. We then argue that the modulo mapping [4] (Section II, Definition 3) is in fact divisible. We therefore

achieve a retransmission mapping with length $O(f^2 \log^2 n)$ for a total schedule length of $n + O(f^2 \log^2 n)$

First, consider a simple scenario to see why 1-good mappings are useful — say all f errors occur during the identity mapping and none during the retransmission mapping.

Observation 1. *If no errors occur during the retransmission mapping, then using a* 1*-good mapping for subset size* f *as the retransmission mapping is a sufficient to transmit all messages.*

To see why this is true, recall that, by definition of 1-good mappings, for any subset S of f messages (and therefore the subset of messages which experienced medium errors during the identity mapping), there will be at least one slot for each of these messages in the 1-good mapping where they do not collide with other messages in S. Therefore, all f messages will transmit without collisions.

However, in the general case, when there may be errors during the retransmission phase, an arbitrary 1-good mapping is not sufficient. To see this, observe that the identity mapping is a 1-good mapping. However, if we have an identity mapping, followed by another identity mapping for retransmission, we cannot guarantee fault tolerance. More generally, in order to guarantee fault tolerance, each message must be assigned to at least f slots in the retransmission mapping ($f + 1$ slots when we include the initial identity mapping); otherwise, every slot containing the message may have a medium error and the message cannot transmit. Therefore, just relying on a mapping being 1-good is not sufficient for fault tolerance. One option is to use multiple 1-good mappings, but that would increase the overhead (length of the retransmission phase). Instead, we argue that the modulo mapping has a much stronger property that makes a single modulo mapping sufficient.

A. Divisible α-good mapping

We define divisible mappings specifically for the case of 1-good mappings. It turns out this definition generalizes for any $\alpha \leq 1$. However, since we only use this property for 1-good mappings, we only define it for $\alpha = 1$.

Definition 5 (Divisible 1-good mappings)**.** *Given* n *messages, a* 1*-good mapping for subset size* m *is divisible if it can be decomposed into* m *phases each containing a contiguous set of slots such that the following condition holds. For every positive integer* $k \leq m$*, every grouping of* k *consecutive phases constitutes a* 1*-good mapping for subset size* k*. In other words, for every size-*k *subset* S *of messages and any group of* k *consecutive phases, there is a slot in the phases to which* x *and no other message in* S *is assigned.*

Divisibility is a counter-intuitive property. The idea is that a divisible mapping is made up of m phases and as we compose these phases, we get divisible mappings for larger and larger subsets. Each phase, in itself, is a 1-good mapping for a subset of size $k = 1$ — that is, if we only have 1 message left, then one phase is sufficient to transmit this one message. Such mappings are trivial — a single slot with all n messages mapped to it is 1-good for all subsets of size 1. But then the definition gets stricter. It says that if we look at all the slots from any $k = 2$ contiguous phases together, they form a 1-good mapping for all subsets of size 2 — that is, for any pair of messages, there are slots within these phases where these messages do not conflict with each other. We continue in this vein for all values of k.

Note that an arbitrary 1-good mapping is not necessarily divisible. For example, the following identity mapping is 1-good for $n = 4$ and $m = 4$:

$$\left\langle \{X_1\}\{X_2\}\{X_3\}\{X_4\} \right\rangle ,$$

since each message has an exclusive slot. However, the mapping is not divisible since we cannot split the mapping into smaller parts such that each part is 1-good for a smaller m.

B. Divisible 1-good mappings guarantee fault-tolerance

It is not at all clear how one would design such mappings and we will show that the modulo mapping is divisible in Section III-C. This subsection argues that a divisible 1-good mapping provides fault-tolerance.

Before getting to the main result, we start by formalizing a generalization of Observation 1.

Lemma 3. *Suppose that there are* η *unsent messages, and a 1-good mapping for subset size* η *is performed. Let* ϕ *be the number of medium errors suffered during the mapping. Then at most* $\min(\eta, \phi)$ *unsent messages remain at the completion of the mapping.*

Proof. Let U, with $\eta = |U|$, be the set of unsent messages at the beginning of the mapping. Clearly once a message has been sent, it cannot become unsent, so the number of unsent messages that remain at the end is at most η.

We focus here on showing that the number of unsent messages that remain at the end is also at most the number of errors ϕ. Consider any particular message $u \in U$. By definition of a 1-good mapping, there exists at least one slot s_u to which u is assigned and to which no other message in U is assigned, which we call a ***collision-free slot for*** u. Let s_u be any one such collision-free slot for u. If no medium error occurs in slot s_u, then u successfully transmits. If a medium error does occur in slot s_u, then we say that u is blocked. Since the chosen collision-free slots are, by definition, distinct for each unsent message (i.e., $s_u \neq s_v$ for $u \neq v \in U$), each medium error blocks at most one message in U. Because only blocked messages may remain at the end, the number of messages that remain is thus at most the number of medium errors ϕ. □

Theorem 1. *Consider a schedule comprising the identity mapping followed by a divisible* 1*-good mapping for subset size* f*. If at most* f *medium errors occur in total, then all messages are successfully sent by the conclusion of the schedule.*

Proof. Let $e_0 \leq f$ be the number of medium errors that occur during the identity mapping. The identity mapping is a 1-good mapping for subsets of size n, so by Lemma 3, the number of unsent messages that remain at the end of the identity mapping

is at most e_0. (In fact, the number of remaining messages is exactly e_0, but we just need an upper bound.)

We now partition the divisible 1-good mapping into some number of rounds starting the numbering from round 1, and we call the identity mapping round 0. This partition is designed in an online fashion as follows. Let n_i be the number of messages that remain at the end of round $i-1$. If $n_i = 0$, then the process terminates. Otherwise, round i corresponds to the next n_i phases of the divisible 1-good mapping. To be well-defined, we must ensure that enough phases remain in the divisible mapping, which we revisit in the next paragraph. Notice that by definition of divisibility, the round i is 1-good for subset size n_i. Let e_i be the number of medium errors occurring during the round. Then by Lemma 3, the number of unsent messages that remain at the end of round i is at most $n_{i+1} \leq e_i$. (In fact, we have $n_{i+1} \leq \min(n_i, e_i)$, but we only need to leverage the weaker claim here.)

It remains to show that the process successfully terminates (i.e., we reach $n_i = 0$ for some i) before we run out of phases in the divisible mapping. Suppose for the sake of contradiction that the process does not successfully terminate, meaning that there exists some earliest round r for which not enough phases remain in the divisible mapping. There are f phases in the divisible mapping, so by the nontermination assumption we have that $\sum_{i=1}^{r} n_i > f$. Applying the fact that $n_i \leq e_{i-1}$ as proved in the previous paragraph, we have $f < \sum_{i=1}^{r} n_i \leq \sum_{i=1}^{r} e_{i-1} = \sum_{i=0}^{r-1} e_i$, i.e., the total number of medium errors incurred through rounds $0, 1, \ldots, r-1$ is strictly more than f. This contradicts the assumption that a total of at most f medium errors occur, and thus the assumption that the process does not terminate is false. □

C. Modulo Mapping is divisible

Now that we know a divisible mapping is useful, we will argue that the Modulo mapping described in Section II from Bender et al. [4] is an instance of a divisible 1-good mapping. The first part of the proof is very similar to the argument in Bender et al. [4] where they prove that it is a 1-good mapping. We will then extend this argument to prove that the same property also implies divisibility.

Theorem 2. *The Modulo mapping (definition 3) is a divisible* 1*-good mapping.*

Proof. Consider a pair of messages j and ℓ, $j \neq \ell$, and suppose they collide in collection p_i. Then it must be the case that $j \equiv \ell \pmod{p_i}$, i.e., their difference is a multiple of p_i. Now consider multiple collections. In particular, let $C_0 = \lceil \log n / \log(m \log n) \rceil$. We argue that in total, j and ℓ cannot collide in C_0 (or more) collections. Suppose for the sake of contradiction that j and ℓ collide in at least C_0 collections. Then their difference must be the product of at least C_0 different primes, all with value at least $m \log n$. Thus $|j - \ell| \geq (m \log n)^{C_0} \geq n$. But the maximum difference between IDs is $n-1$, which gives us a contradiction. Thus, j and ℓ must collide fewer than C_0 times.

Bender et al. [4] used the same observation to prove the mapping 1-good. We use this property to prove divisibility. Let C_0 successive collections constitute a phase. Since the mapping contains $C = m\lceil \log n / \log(m \log n) \rceil$ collections, there are $C/C_0 = m$ phases. Given an integer k ($k \leq m$), consider any size-k subset S of messages, and consider any particular message j in this set. By the logic above, each other message collides with j in fewer than C_0 collections. If we sum across the collisions with all $k-1$ other messages in S, in total j must experience fewer than $(k-1)C_0$ collisions with other messages from S. Thus, for any k contiguous phases and hence kC_0 contiguous collections, message j must have at least one collection in which it does not experience a collision. Therefore, the slots of any k phases (each phase contains C_0 collections) make a 1-good mapping for a subset of size k for all $k < m$, which is the definition of divisibility. □

Putting everything together, the retransmission phase consists of a modulo mapping, generated according to Definition 3 with parameter $m = f$. By Lemma 1, for $m = f$ this mapping has length $O(f^2 \log^2 n)$. By Theorem 2, this mapping is divisible. By Theorem 1, a divisible 1-good mapping is sufficient. We thus conclude with the following:

Corollary 2.1. *Consider the schedule consisting of the identity mapping followed by the modulo mapping for subset size f. If at most f medium errors occur, then all messages are successfully sent by the conclusion of the schedule. Moreover, the retransmission length (i.e., the overhead) is $O(f^2 \log^2 n)$.*

IV. Generating fault-tolerant mapping with reducible α-good mappings

This section applies multiple α-good mappings to produce a shorter fault-tolerant schedule. In order to apply the method described in this section, we will define a property of α-good mappings, namely ***reducibility***. Once we define this property, we will show that the algorithm given here works for any reducible α-good mapping, including the 1-good mapping. The main advantage is achieved when $\alpha < 1$, which allows for a shorter mapping length and hence better overall schedule. As a simple extension of Bender et al. [4], we show that there exists a reducible $1/2$-good mapping for m with length $\Theta(m \log n)$. Coupled with the main algorithm in this section for producing a good retransmission mapping from α-good mappings, this gives us a retransmission mapping length of $O(f \log f \log n)$ for an overall schedule length of $n + O(f \log f \log n)$.

This bound is asymptotically better than the schedule produced using divisible 1-good mappings. It should be noted, however, that we do not know of any efficient algorithm for producing a reducible $1/2$-good mapping. Rather the simple random ball-bin method is likely to generate a $1/2$-good mapping, but *verifying* that the resulting mapping is actually $1/2$-good is computationally expensive. Nevertheless, a mapping for each set of parameters need only be generated once — therefore, we could use it to get small schedules.

A. Reducible α-good mapping

Recall that an α-good mapping for subset size m only guarantees that for subsets of size m, at least αm of the message are assigned to time slots without collision. What if there are only $k \ll m$ unsent messages still trying to send? It turns out that the α-good property as defined does not directly ensure progress.

To understand the issue, consider a specific subset S of size $|S| = k$, $m = 2k$ and $\alpha = 1/2$. One valid (but not very effective) $1/2$-good mapping would be a mapping that assigns every element of S to the same single slot, and all other $n - k$ elements to their own individual slots. This mapping meets the definitions of being $1/2$ good for subsets of size $m = 2k$, as all such subsets must include at least k elements assigned to their own slots. However, if we only have k unsent messages corresponding exactly to the subset S, then none of the messages may be collision free.[2]

For example, consider the following mapping is $1/2$-good for $n = 8$ and $m = 8$.

$$\left\langle \{X_1\}\{X_2\}\{X_3\}\{X_4\}\{X_5, X_6, X_7, X_8\} \right\rangle .$$

In the mapping, 4 messages — X_1, X_2, X_3, X_4 have their own slots. However, this mapping is not $1/2$-good for $n = 8$ and $m = 4$ since all the messages in the subset X_5, X_6, X_7, X_8 collide with each other. It is also not $1/2$-good for many other values of m.

To overcome this issue, we extend the property to also be *reducible*, which simply means that the mapping needs to be α-good for subsets of size at most m, not just exactly equal to m. The preceding example is not reducible.

Definition 6 (Reducible α-good mapping)**.** *Given n messages, a **reducible α-good mapping for subset size m** is a mapping that is α-good for all subset sizes $k \leq m$.*

This property guarantees that if there are $k \leq f$ unsent messages, a reducible α-good mapping for $m = f$ will send at least αk messages without collisions (in the absence of medium errors). We note here that 1-good mappings are always reducible.

Lemma 4. *Any 1-good mapping is reducible.*

Proof. Consider a 1-good mapping for subset size m. We must show that it is also 1-good for $k < m$. In particular, for a subset S with $|S| = k$, we must show that every element in the subset has a collision free slot. Let S' be any subset with $|S'] = m$ and $S \subset S'$, i.e., augment S by adding any $m - k$ elements. By definition of 1-good, every element in S' has a collision-free slot, and hence so does every element in S. □

However, since we already have a retransmission schedule which contains only one 1-good mapping, we will focus here on reducible mappings for $\alpha < 1$ in order to get shorter retransmission mappings overall.

[2]This is a problem with the definition of α-good mapping. We will see in Section IV that the mapping from [4] has a stronger property of reducibility.

B. Achieving fault-tolerance by repetitions

This section gives two methods that integrate multiple reducible α-good mappings to generate a retransmission mapping. We will argue that the schedule consisting of an identity mapping followed by this retransmission mapping guarantees fault tolerance. The overall schedule length depends on the sizes of the α-good mappings. Section IV-C proves that small $1/2$-good reducible mappings exist and analyzes the overall schedule length.

We begin by defining a ***geometric schedule*** (shown in Algorithm 1) — so called since the subset size for subsequent mappings decreases geometrically. For this schedule, we will prove that it achieves fault tolerance.

Algorithm 1 Constructing a retransmission mapping from reducible $1/2$ good mappings using the geometric method

```
▷ I(n) is the identity mapping for n messages
▷ M(n, m) is a reducible α-good mapping for subset size m.
function GEOMETRIC-INTEGRATION(n, f, α)
    schedule = I(n)
    for i ← 0 to log f do
        append ⌈((1/α)(1 + 2^(i+1))⌉ copies of M(n, f/2^i) to schedule
    end for
    return schedule
end function
```

The schedule comprises the identity mapping followed by the retransmission mapping, which contains several reducible α-good mappings for different subset sizes m. Consider an instance on n messages, f total medium errors, and goodness parameter α. The retransmission mapping comprises roughly $\log f$ phases (logarithms are base 2), where each phase corresponds to an iteration of the main loop in Algorithm 1. The ith phase consists of $\lceil (1/\alpha)(1 + 2^{i+1}) \rceil$ (roughly $\Theta(2^i)$) repetitions of a reducible mapping that is α-good for subset size $f/2^i$. In each phase, the number of mappings doubles, but the length of each repetition of the mapping halves.

To prove correctness, the main goal is to prove the following invariant: at the start of phase i, there are at most $f/2^i$ unsent messages. This invariant is necessary — if more unsent messages exist, then a mapping that is α-good for $m = f/2^i$ has no guarantees on collisions for subsets of size larger than m. Moreover, if the invariant holds, then there are no unsent messages at the end of all the iterations. This invariant, or more precisely the inductive step of the invariant, is captured by the following lemma.

Lemma 5. *Suppose that the ith phase (iteration i) begins with at most $f/2^i$ unsent messages. Suppose also that at most f medium errors occur during the phase. Then at the end of the phase, there are at most $f/2^{i+1}$ unsent messages.*

Proof. Suppose, for the sake of contradiction, that a phase fails, i.e., ends with at least $f/2^{i+1}$ unsent messages. Then

there must be at least that many unsent messages at the start of every repetition of the α-good mapping. We shall show that with only f medium errors, keeping this many messages alive would require more than f medium errors, hence generating the contradiction.

Each reducible α-good mapping in the phase is good for subset sizes up to $f/2^i$. Since by assumption there are at least $f/2^{i+1}$ unsent messages throughout, in each mapping repetition there must be at least $\alpha f/2^{i+1}$ messages that are assigned to at least one collision-free slots. For each message, consider just one such collision free slot. These messages would successfully transmit as long their slot does not suffer a medium error. That is to say, every such collision-free slot that does not incur a medium error results in a message being successfully transmitted. Note that this number of collision-free messages depends only on the assumption that there are still a lot of unsent messages — it does not depend on how medium errors are assigned in previous repetitions.

Summing across all repetitions, the total number of collision-free slots is at least $\alpha f/2^{i+1}$ times the number of repetitions, or at least $(1/\alpha)(1+2^{i+1})\cdot\alpha f/2^{i+1} = f+f/2^{i+1}$ collision-free assignments. If $\hat{f}$ medium errors occur, the number of successful transmission is thus at least $f + f/2^{i+1} - \hat{f}$. To conclude the proof, we observe that the phase begins with at most $f/2^i$ unsent messages by assumption, and hence there must be fewer than $f/2^{i+1}$ successful transmissions to keep the surviving number of unsent messages above $f/2^{i+1}$. Thus we have $f+f/2^{i+1}-\hat{f} < f/2^{i+1}$ or $\hat{f} > f$, which contradicts the assumption that there are only f medium errors in the phase. □

Note that the preceding lemma is stronger than necessary in that it allows for f medium errors in each phase, whereas there are only f medium errors in total.

Theorem 3. *Consider the schedule generated by the algorithm described in Algorithm 1 for a particular settings of n, f, and $\alpha > 0$ for which the specified mappings exist. If at most f medium errors occur, then all n messages successfully transmit by the end of the schedule.*

Proof. Via induction over phases, we can show that at the start (and end) of iteration i, there are at most $f/2^i$ (and $f/2^{i+1}$) unsent messages. For the base case, at the beginning of the retransmission mapping (start of phase $i = 0$) there are at most $f = f/2^0$ unsent messages. For each subsequent phase, Lemma 5 shows the inductive step. Thus, after $\log(f)$ phases, there is $\leq 1/2$ unsent message i.e., no unsent messages. □

For the types of mappings we know, there are not many interesting values of α—just $\alpha = 1$, which gives an easy constructive mapping, and $\alpha = 1/2$, which gives a much smaller mapping. Pushing α smaller may result in the α-good mapping being smaller, but this mapping-size improvement is offset by increasing the number of repetitions with $(1/\alpha)$.

Nevertheless, it is possible to improve some constant factors in the integration algorithm itself, albeit at a cost of complicating the proof. We thus chose to lead with the algorithm with the clearer proof in this section. Algorithm 2 gives an alternative ***harmonic algorithm*** — so called since the subset size decreases harmonically in each iteration — specifically for $\alpha = 1/2$, which decreases the overall number of repetitions at each subset size. This is the version we use in our experiments. The main loop structure is different in this algorithm: there are now f phases, where phase i now corresponds to subset size $m = f/i$, but the advantage is that each phase now has just a constant number of repetitions. Though quite different at first glance, it's worth noting that the algorithms and proofs are conceptually similar: as with the geometric sequence, the harmonic sequence includes $\Theta(2^i)$ numbers that are between $1/2^i$ and $1/2^{i-1}$.

Algorithm 2 A modified (harmonic) algorithm for constructing a retransmission mapping using reducible $1/2$-good mappings.

```
▷ I(n) is the identity mapping for n messages
▷ M(n, m) is a reducible 1/2-good mapping for subset size m.
function HARMONIC-INTEGRATION(n, f)
    schedule = I(n)
    for i ← 1 to f do
        append 3 copies of M(n, f/i) to schedule
    end for
    return schedule
end function
```

The key to proving correctness of this harmonic mapping is to show that iteration i begins with at most f/i, and ends with at most $f/(i+1)$, unsent messages. This property is analogous to Lemma 5, and it is necessary and sufficient for similar reasons. That is, the mapping applied during iteration i is only good for subsets of size at most f/i, so to argue anything about the number of collision-free messages we need to have at most f/i unsent messages. Moreover, at the end, having $f/(f+1) < 1$ unsent messages implies that there are no unsent messages.

Our proof of this key invariant follows a somewhat different structure from the analogous argument for the geometric version. To understand the setup, it is best to consider what would happen if there were no medium error — each invocation of a $1/2$-good mapping (for sufficiently large subset size) would reduce the number of unsent messages by $1/2$, which would mean $1/8$ per iteration. Thus, in the absence of medium errors, the number of unsent messages at the start of iteration i would be $f/8^{i-1}$, which is far lower than the harmonic guarantee we set out to achieve. The point is that to keep the number of unsent messages as high as f/i in every iteration, the number of medium errors must already be quite large.

Our analysis thus somewhat decouples the iteration number from the number of messages remaining. Our goal is to characterize each phase according to how many steps it takes in the harmonic sequence. For example, going from $f/2$ unsent messages down to $f/5$ takes three steps (the step from $f/2$ to $f/3$, the step from $f/3$ to $f/4$, and the step from $f/4$

to $f/5$) in the harmonic sequence. To this end, we define the ***harmonic rank*** over positive integers $1, 2, \ldots, f$, denoted by $hrank(x)$, to be the value $j = hrank(x)$ such that $f/(j+1) < x \leq f/j$. For example, for $f = 16$, we have $hrank(10) = 1$, $hrank(7) = 2$, and $hrank(4) = 4$.

As before, we call iteration i of the main loop ***the ith phase***, which now comprises three $1/2$-good mappings for subset size f/i. As noted above, without medium errors the harmonic rank should increase by more than 1 in each phase. If there are some medium errors, the harmonic rank may increase by less. The following lemma flips this dependence around — the change to the harmonic rank gives us a bound on the number of medium errors.

Lemma 6. *Let u and u' denote the number of unsent messages at the start and end of a particular phase, respectively. Suppose that $u \leq f$ and that $u' \geq 1$, let $h = hrank(u)$, and let $h' = hrank(u')$. Suppose also that the phase number i satisfies $i \leq h$, or equivalently $u \leq f/i$. Then the number of medium errors that occurred during the phase is strictly greater than $2f/(h'+1) - f/(2h)$.*

Proof. Because by assumption $u \leq f/i$, each copy of the $1/2$-good mapping during this phase is $1/2$-good for all $m \geq u$. It follows that throughout the phase, for each copy of the mapping, at least half of the remaining unsent messages are each assigned to at least one collision-free slot.

We now consider each of the copies of the $1/2$-good mapping. Let u_k, for $k \in \{1, 2, 3\}$ denote the number of unsent messages that remain after performing the kth repetition of the $1/2$-good mapping in this phase. Similarly, let x_k denote the number of medium errors occurring during copy k. Our goal is to prove that $x_1 + x_2 + x_3 > 2f/(h'+1) - f/(2h)$. To start, observe that

$$\begin{aligned} u_1 &\leq u/2 + x_1, \\ u_2 &\leq u_1/2 + x_2, \quad \text{and} \\ u_3 &\leq u_2/2 + x_3 \ . \end{aligned}$$

These inequalities follow from the fact that each unsent message assigned to a collision-free slot only remains unsent if it suffers a medium error.

Combining the three inequalities, we have

$$\begin{aligned} &u_1 + u_2 + u_3 \leq x_1 + x_2 + x_3 + u/2 + u_1/2 + u_2/2, \text{or} \\ &x_1 + x_2 + x_3 \geq u_3 + u_1/2 + u_2/2 - u/2 \end{aligned}$$

Since the number of unsent messages is monotonically decreasing, we have $u_i \geq u' > f/(h'+1)$. We also have $u \leq f/h$. Substituting these back into our inequality yields

$$x_1 + x_2 + x_3 \geq 2u' - u/2 > \frac{2f}{h'+1} - \frac{f}{2h} \ .$$

□

Observe that if the harmonic rank increases by a lot during the phase (for instance, if $h' \geq 4 \times h$), then Lemma 6 does not say anything about the number of medium errors — in this case $2f/(h'+1) - f/(2h)$ is negative. On the other hand, a slower increase to the harmonic rank can only occur in the presence of medium errors.

The next lemma shall be useful to bound the number of errors for a sequence of k phases starting with harmonic rank h_0 and ending with harmonic rank h_k. Roughly speaking, the implication is that the lowest number of errors necessary[3] to keep x messages alive across multiple phases is realized by allowing all but x messages to succeed immediately in the first phase, and then using all medium errors to maintain x messages in the subsequent phases.

Lemma 7. *Consider a monotonically increasing sequence of $1 \leq h_0 \leq h_1 \leq \cdots \leq h_k$. Then we have*

$$\sum_{i=1}^{k} \left(\frac{2}{1+h_i} - \frac{1}{2h_{i-1}} \right) \geq \frac{k+1}{1+h_k} - \frac{1}{2h_0} \ .$$

Proof. We start by splitting the sum into

$$\begin{aligned} &\sum_{i=1}^{k} \left(\frac{2}{1+h_i} - \frac{1}{2h_{i-1}} \right) \\ &\quad = \sum_{i=1}^{k} \frac{1}{1+h_i} + \sum_{i=1}^{k} \left(\frac{1}{1+h_i} - \frac{1}{2h_{i-1}} \right) \\ &\quad \geq \sum_{i=1}^{k} \frac{1}{1+h_k} + \sum_{i=1}^{k} \left(\frac{1}{1+h_i} - \frac{1}{2h_{i-1}} \right) \\ &\quad = \frac{k}{1+h_k} + \sum_{i=1}^{k} \left(\frac{1}{1+h_i} - \frac{1}{2h_{i-1}} \right) \end{aligned}$$

To complete the proof, we simply observe that for $h_i \geq 1$, we have $1/(1+h_i) \geq 1/(2h_i)$, or equivalently $1/(1+h_i) - 1/(2h_i) \geq 0$. We can thus cancel all consecutive intermediate terms in the sum (the sum telescopes), and we are left with

$$\sum_{i=1}^{k} \left(\frac{1}{1+h_i} - \frac{1}{2h_{i-1}} \right) \geq \frac{1}{1+h_k} - \frac{1}{2h_0} \ .$$

□

We are now ready to prove the main correctness invariant: that each phase i starts and ends with at most f/i and $f/(i+1)$ unsent messages, respectively, and hence the mapping is good for the number of remaining unsent messages.

Lemma 8. *Let u_{i-1} and u_i denote the number of unsent messages at the start and end, respectively, of phase i. If at most a total of f medium errors occur (over the whole schedule), then for all i, $0 \leq i \leq f$, we have $u_i \leq f/(i+1)$.*

Proof. Suppose for the sake of contradiction that there exists an i such that $u_i > f/(i+1)$. Let $x = \min\{i | u_i > f/(i+1)\}$ be the first such phase number.

Note that u_i is always an integer, and $f/(i+1) > 0$, so $u_i \geq 1$ for all $i \leq x$. Moreover, we also have $u_i \leq f$ due to

[3]This statement is with respect to the lower bound of Lemma 6, which allows a negative number of errors. That lower bound is thus looser than the reality, where the number of medium errors cannot be negative.

Lemma 3, which applies because the identity mapping is 1-good and suffers at most f medium errors. Thus for $0 \le i \le x$, we have $1 \le u_i \le f$, and the harmonic rank is well-defined for all i in this range. We thus let $h_i = hrank(u_i)$ for $0 \le i \le x$.

By choice of x, we have $u_{i-1} \le f/i$ for all $i \le x$. Thus the conditions of Lemma 6 are met for all $i \le x$. (We map each u_{i-1} and u_i to u and u', respectively, in the statement of that lemma.) Following from Lemma 6, the total number of medium errors during phases 1 through x inclusive is strictly greater than

$$\sum_{i=1}^{x} \left(\frac{2f}{1+h_i} - \frac{f}{2h_{i-1}} \right) = f \sum_{i=1}^{x} \left(\frac{2}{1+h_i} - \frac{1}{2h_{i-1}} \right) .$$

Applying Lemma 7,

$$f \sum_{i=1}^{x} \left(\frac{2}{1+h_i} - \frac{1}{2h_{i-1}} \right) \ge f \left(\frac{x+1}{1+h_x} - \frac{1}{2h_0} \right) .$$

If u_0 messages survive the identity mapping, then the number of medium errors during the identity mapping is u_0 (Lemma 3). We have $u_0 > f/(h_0+1)$ by definition of the harmonic rank. Adding all the errors together, the total number of medium errors is strictly greater than

$$f \left(\frac{x+1}{1+h_x} - \frac{1}{2h_0} \right) + \frac{f}{1+h_0} \ge \frac{f(x+1)}{1+h_x} ,$$

which follows because $h_0 \ge 1$.

By choice of x, $u_x > f/(x+1)$ and hence $h_x = hrank(u_x) < x+1$, or $h_x \le x$. We thus have a strictly more than

$$\frac{f(x+1)}{1+h_x} \ge \frac{f(x+1)}{1+x} = f$$

medium errors. This generates a contradiction as the number of medium errors is assumed to be at most f and cannot be strictly more than f. □

Theorem 4. *Consider the schedule generated by the algorithm described in Algorithm 2 for a particular settings of n and f. If at most f medium errors occur, then all n messages successfully transmit by the end of the schedule.*

Proof. Lemma 8 implies that $u_f \le f/(f+1) < 1$ — the number of unsent messages at the end of phase f is 0. Thus, all messages must successfully transmit by the schedule. □

C. Applying reducible $1/2$-good mappings of small size

We have shown how to build fault-tolerant schedules via integrating many reducible α-good mappings. In this subsection, we present a $1/2$-good mapping that is reducible and hence applies to achieve fault tolerance.

The mapping is the same mapping described by Bender et al. [4]. We are not claiming the mapping itself as a contribution, but we present some of the analysis for completeness. Bender et al. [4] use the probabilistic method on random assignments to argue that $1/2$-good mappings of modest size exist. That is, they show that a random process for generating a mapping has some chance of producing a $1/2$-good mapping. Thus, one can obtain a $1/2$-good mapping by repeating the random process and testing the result.

Bender et al. [4] omit several of the proofs from their paper, however, so we include those details here. Moreover, we show that the same mapping is also reducible.

The following lemma is similar to one given by Bender et al. [4], but no proof is provided in their paper. This balls-in-bins lemma is the main crux of the mapping: the full mapping comprises $\Theta(\log n)$ uniformly random (balls-in-bins) assignments of this form.

Lemma 9. *Let $18m$ denote the number of bins and $k \le m$ be the number of balls. There exists a constant $\delta \in (0,1)$ such that: if the k balls are thrown uniformly at random into the $18m$ bins, then we have* $\Pr($*fewer than $k/2$ bins have exactly one ball*$) < \delta^k$.

Proof. Let cm be the number of bins (where we shall derive the $c \ge 18$ in the proof). Consider tossing k balls, $k \le m$, into those bins. Let ρ denote the number of bins containing at least 1 ball. Let x denote the number of bins having at least 2 balls. We want to bound the number $y = \rho - x$ of bins that contain exactly 1 ball. We have $(\rho - x) \times 1 + x \times 2 \le k$, which leads $x \le k - \rho$. Thus, we have

$$y \ge 2\rho - k$$

We want $y \ge 2\rho - k \ge k/2$. Observe that $2\rho - k \ge k/2$ is equivalent to $\rho \ge 3k/4$. It follows that if $\rho \ge 3k/4$, then $y \ge 2\rho - k \ge k/2$. Thus, we have $\Pr[y \ge k/2] \ge \Pr[\rho \ge 3k/4]$ which leads

$$\Pr[y < \frac{k}{2}] \le \Pr[\rho < \frac{3k}{4}]$$

Now, we want to show $\Pr[\rho < 3k/4] < \delta^k$. For any subset of $[1,i]$ bins where $i \le m$, we have the probability p that all balls land in those bins is $p \le (\frac{i}{cm})^k$. The number of subsets of $[1,i]$ bins is $\binom{cm+i-1}{i} < e^i (\frac{i}{(c+1)m})^{-i}$. Apply a union bound over all the subsets, we have $\Pr[\rho < i] < \Pr[\rho \le i] \le e^i (\frac{i}{(c+1)m})^{-i} (\frac{i}{cm})^k$. In particular, for $i = 3k/4$, we have

$$\Pr\left[\rho < \frac{3}{4}k\right] < \left(\left(\left(\frac{3e^3}{4c} \right) \left(\frac{c+1}{c} \right)^3 \right)^{\frac{1}{4}} \right)^k \qquad (1)$$

Select $\delta = ((\frac{3e^3}{4c})(\frac{c+1}{c})^3)^{\frac{1}{4}}$, and notice that for $c \ge 18$, we have $\delta < 1$. Above all, we have

$$\Pr\left[y < \frac{k}{2}\right] \le \Pr\left[\rho < \frac{3k}{4}\right] < \delta^k . \qquad (2)$$

□

We now show that the balls-in-bins method has a non-zero chance of producing a reducible $1/2$-good mapping. Therefore, a reducible $1/2$-good mapping must exist.

Theorem 5. *The balls-in-bins method (definition 4) produces a reducible $1/2$-good mapping with at least a constant probability.*

Proof. The method generates $\Theta(\log n)$ independent collections. For each collection, we randomly (uniformly) allocate n messages to $\Theta(m)$ slots. Due to Lemma 9, given k ($k \leq m$) messages, the probability P_1 that fewer than $k/2$ slots that each slot has exactly one of the k messages is $P_1 < \delta^k$ for some $\delta \in (0,1)$. Since we have $\theta(\log n)$ independent collections. Thus, we have the probability P_2 that every collection has less than $k/2$ slots that each slot has exactly one of the k messages is $P_2 = P_1^{d \log n} < (\delta^k)^{d \log n}$ for $d \geq 1$. Since there are $\binom{n+k-1}{k} \leq \binom{2n}{k} \leq (\frac{2en}{k})^k \leq e^k 2^{k \log n}$ ways to pick $[1,k]$ messages out of n messages. We apply a union bound over all possible subsets that contain $[1,k]$ messages, we have the probability P_3 that the mapping is reducible $1/2$-good for subset size k is $P_3 < e^k 2^{k \log n} \delta^{dk \log n} < 1$ for sufficiently large d. Thus, it is possible to get a reducible $1/2$-good mapping for subset size m via the probability method. □

Now that we have shown that reducible $1/2$-good mappings of length $O(m \log n)$ exist, we can now compute the total length of the the retransmission mapping created using the method in Algorithm 1 or Algorithm 2.

Lemma 10. *Assuming we generate reducible $1/2$-good mappings using balls-in-bins method and then use the above methods to generate a retransmission mapping, the size of the retransmission mapping in either case is $O(f \log f \log n)$*

Proof. For the geometric method shown in Algorithm 1, we have $\log f$ phases. Phase i contains 2^{i+1} copies of a reducible mapping for subset size $f/2^i$. Therefore, the length of phase i is $O(2^{i+1} f/2^i \log n) = O(f \log n)$. Therefore, the total length is $O(f \log f \log n)$.

For the harmonic method shown in Algorithm 2, we have f phases and each phase contains 3 copies of a reducible mapping for subset size f/i. Therefore, the length of each phase is $O(f/i \log n)$. If we add over all i, we get $O(\sum_{i=1}^{f} f/i \log n) = O(f \log f \log n)$. □

Putting everything together, by Theorem 5, reducible $1/2$-good mappings exist. By Theorem 3, we can compose these mappings into a retransmission mapping which guarantees fault tolerance for f errors and the length of this retransmission schedule is $O(f \log f \log n)$ (Lemma 10 for the total schedule length of $n + O(f \log f \log n)$).

V. Evaluation

This section describes our analytical evaluation of the overhead due to fault tolerance using various methods for generating the retransmission mapping. The methods described in this paper are have asymptotically better overheads than previous methods — sublinear in n instead of $O(nf)$. Until n or f grow large, however, the constants involved are of practical concern. We now try to understand at what values of n and f these new methods begin to become effective.

We compare the lengths of the retransmission schedules. Our baseline is the dual mapping [3], henceforth denoted as DUAL. In DUAL, every message appears in f slots and each slot contains at most 2 messages. Let $|M|$ denote the length of mapping M. From [3], we have $|\text{DUAL}| = nf/2$. We first compare DUAL with our approach based on the modulo mapping of Section III, denoted by DIV for "divisible." We then compare DUAL against our second approach in Section IV. We use RDC, for "reducible," to denote the retransmission schedule that results from generating $1/2$-good mappings with the ball-bin method and the harmonic integration.

A. Divisible 1-good mapping

The modulo mapping lends itself to a simple algorithm for constructing a divisible 1-good mapping. We build the DIV mapping following the specification in Definition 3 — that is, we simply identify the smallest $f \log n / \log f$ primes and then use these primes to construct each collection by applying the appropriate modulo operation to each message ID.

We begin by presenting the exact length of DIV for different n's and f's. We observe that DIV outperforms the baseline DUAL when the ratio n/f is sufficiently large, as expected by the asymptotic bound. We then determine the minimum value of n/f for which DIV outperforms the baseline.

Figure 1 shows the comparison of DIV against the baseline from different perspectives. For Figure 1a, we try different values of n ranging between 1000 and 10000 in steps of 1000 and values of f between 20 and 100 in steps of 20. For each pair of n and f, we calculate the length of DIV divided by the length of the baseline (lower is better, and the value below 1 indicates DIV outperforms the baseline). To highlight some numbers, when $n = 5000$ and $f = 40$, DIV's length is roughly $2/3$ the length of DUAL. When $n = 10000$ and $f = 40$ or at $n = 5000$ and $f = 20$, the ratio is about $1/3$. Because the asymptotic ratio between the two retransmission mapping lengths is $O(|\text{DIV}|/|\text{DUAL}|) = O(f^2 \log^2 n/(nf)) = O((f/n) \log^2 n)$, these experiments confirm the expected behavior that the ratio between their lengths is roughly inversely proportional to n/f.

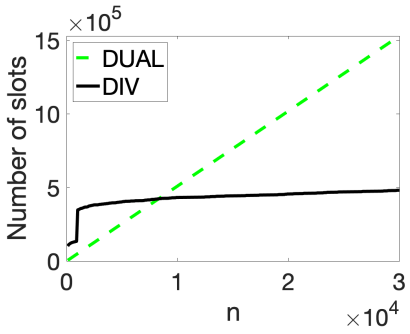
Figure 1b keeps f constant and shows the dependence on n. Again, we see that as n increases, the 1-good mapping gains advantage — the size of the mapping increases very slowly with increasing n compared to the baseline. Thus, in low medium-error environments, DIV scales very well as n increases. In contrast, for fixed f DUAL's length is proportional to n.

Figure 1c tries to understand the performance in a slightly different way. In the above evaluation, we see that when n/f is sufficient large, DIV outperforms the baseline, as is to be expected from the bound. In particular, the asymptotics indicate that DIV (with —DIV— $= O(f^2 \log^2 n)$) should outperform DUAL (with —DUAL— $= nf/2$) when $f = O(n/\log^2 n)$ or equivalently when $n/f = \Omega(\log^2 n)$ for sufficiently large c. We next try to understand how large n/f needs to be to overcome the true constants in that $\Omega(\log^2 n)$ term. Let r_1 be the minimal n/f s.t. $|\text{DIV}| < |\text{DUAL}|$. As n increases, r_1 also increases, but slowly due to the dependence is on $\log^2 n$.

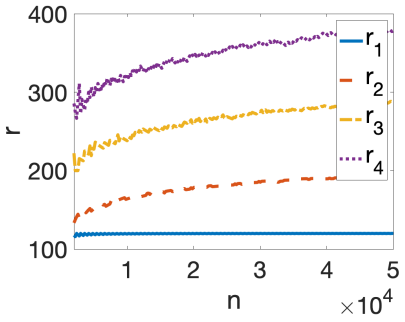
Let r_i denote the minimal n/f s.t $|\text{DIV}| < |\text{DUAL}|/i$. We call i the outperformance level and want to understand at what values of n, we start to see a particular level of advantage from using a divisible mapping rather than the baseline. In

n \ f	20	40	60	80	100
1000	1.306	2.629	4.004	5.475	6.944
2000	0.6647	1.408	2.158	2.914	3.717
3000	0.4665	0.9839	1.496	2.002	2.55
4000	0.3618	0.7516	1.136	1.539	1.951
5000	0.2943	0.6122	0.9261	1.254	1.591
6000	0.2493	0.5192	0.786	1.055	1.341
7000	0.2137	0.4489	0.6818	0.9217	1.158
8000	0.19	0.3928	0.6037	0.8141	1.032
9000	0.1715	0.3551	0.5397	0.7338	0.928
10000	0.1544	0.3196	0.4885	0.6665	0.8445

(a) Normalized mapping length



(b) Mapping length when $f = 100$



(c) The value of r_1, r_2, r_3, r_4

Fig. 1: Numerical results of DIV. Figure 1a presents the ratio of DIV length to DUAL length for $n \in [1000, 10000]$ and $f \in [20, 100]$; Figure 1b keep f constant and shows the sub-linear trend of DIV length as n increases, comparing to the linear trend of DUAL; Figure 1c presents the trend of r_1, r_2, r_3, r_4 as n increases, where r_i denotes the minimal n/f such that the length of DIV is shorter than DUAL length over i. DUAL mapping is the baseline.

n	f	c	d	RDC	DUAL
200	2	1	2	97	200
200	3	1	2	169	300
200	4	1	2	385	400
400	2	1	2	109	400
400	3	1	2	190	600
400	4	1	2	433	800

TABLE I: Comparison of number of slots between RDC and DUAL (baseline) for small n's and f's

the evaluation, we evenly select n in $[2000, 50000]$ (the step is 200). We calculate r_i for each n and the results are in Figure 1c. As n gets bigger, r_i increases with n as expected but very slowly. Therefore, above a certain value of n it might always be useful to use divisible mappings unless f is very large. We can also see that for $n \in [2000, 50000]$, $r_1 \leq 120$, $r_2 \leq 200$, $r_3 \leq 300$, $r_4 \leq 400$. It indicates that DIV can be significantly better than the baseline in a low medium-error environments.

B. *Reducible* $1/2$*-good mappings*

We now consider the retransmission schedule RDC, which is built using the ball-bin method coupled with the harmonic technique in Section IV. These experiments are more complicated as the ball-bin method is not guaranteed to generate a $1/2$-good mapping. Instead, we must randomly construct candidate mappings and check if they are $1/2$-good. This check is prohibitively expensive for large values of n and f since we must check for collisions for all f-size subsets of n messages and there are about n^f of them.

There are some interesting questions to be asked about $1/2$-good mappings, however, including the specific setting of constant parameters c and d in Definition 4. In this section, we will first explore which values of these constants for small values of n and f lead to producing mappings that are actually $1/2$-good. We then compare the (potential) length of $1/2$-good mappings for various values of c and d to the baseline.

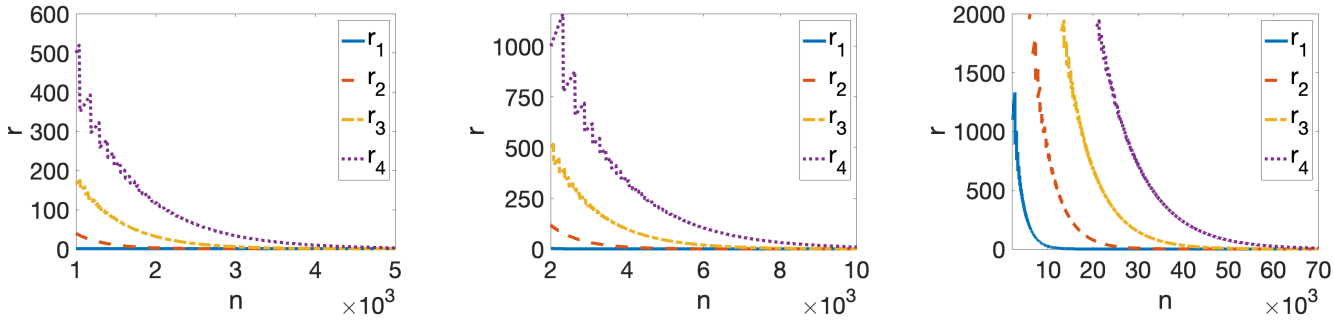
Determining c *and* d *of the reducible* $1/2$*-good mapping in practice:* Recall that RDC is built up with reducible $1/2$-good mappings. We see in Lemma 9 that a constant of $c = 18$ is sufficient. The constant d appears in Theorem 5, and it is also large. (We do not solve for d, merely arguing that a sufficiently large d exists.) The size of reducible $1/2$-good mapping depends on $c \times d$. While the theoretical value of this quantity is large (since they are based on what can be proven using simple mathematical techniques, in practice, we can often get a valid mapping for smaller values.

For small values of n and f, we can try small values of c and d and try to generate $1/2$-good mappings by trying the balls-into-bin method several times. Table I shows the values of c and d that were sufficient to find valid mappings for $n = 200, 400$ and $f = 2, 3, 4$. Note, in the table, that relatively small values of c and d are sufficient to get valid mappings — $c \times d = 2$ is sufficient for all values that we tested. This indicates that although our mathematically proven upper bounds have high constants, the effective values of the true constants may not be that large. The table also shows the length of the smallest correct schedule we were able to find using reducible mappings. We see that even for such small values of f and n, RDC generally uses fewer slots than DUAL for its retransmission mapping, often by a significant amount. This advantage is only likely to increase for large n and f if we can find an efficient way to generate these mappings.

a) Projection of performance of RDC for large n *and* f*:* It is difficult to properly evaluate RDC for large n and f since checking whether a randomly generated mapping is correct is prohibitively expensive. Thus, we can only speculate here.

Figure 2 shows the comparison of length of the retransmission schedule constructed using reducible mappings to the baseline for various guesses for the correct value of $c \times d$. We saw earlier that small values of $c \times d$ is often sufficient, but we do not know if that is also true when n and f are large. Therefore, we show the comparison for various values.

Again, we use the values of r_1, r_2, r_3 and r_4 as defined for divisible mappings earlier. We see that different from the r_i of DIV, the r_i of RDC decreases to 1 when n gets sufficiently large — implying that for large enough n, all values of $f \leq n$



(a) $c \times d = 2$ (b) $c \times d = 4$ (c) $c \times d = 18$

Fig. 2: The trend of r_1, r_2, r_3, r_4 of RDC as n increase, where r_i denotes the minimal n/f such that the length of RDC is shorter than DUAL length over i.

benefit from using reducible mappings. This is also easy to see from the theoretical results — the length of both the baseline and reducible mapping method increase (almost) linearly with f, while divisible 1-good mapping had a quadratic dependence on f. While the divisible mapping was only good for small f and (relatively) large n, this method is better than (at least as good as) the baseline for all values of f for large enough n.

However, the correct values of c and d matters since constants matter in practice. For example, let us focus on r_3 in Figure 2 — $r_3 = 1$ shows the value of n at which RDC is better than DUAL by a factor of 3. Roughly, when $c \times d = 2$, $r_3 = 1$ if $n \geq 4000$. When $c \times d = 4$, $r_3 = 1$ if $n \geq 8000$. When $c \times d = 18$, $r_3 = 1$ if $n \geq 60000$. Therefore, the smaller the value of $c \times d$, the smaller the value of n at which we should prefer to use RDC over DUAL for all values of f. However, even when $c \times d$ is large, there still exists a sufficient large n such that RDC always outperforms the baseline. It is also worth noting that RDC is significantly better than DIV even for large $c \times d$ for reasonable values of n.

VI. Related Work

We now discuss some related work on both contention resolution and fault tolerance.

Contention resolution technologies such as randomized backoff protocols are widely used to resolve contention for simple multiple access channels. In a randomized backoff protocol, when multiple jobs try to access a simple channel at the same time, they collide with each other and both back off for a random amount of time before retrying. Jobs can arrive at the simple channel following the statistical queuing-theory model [5] or in bursty (all jobs arrive at time 0) [6]. Jobs can have unit-sizes [6], or heterogeneous sizes [4]. Jamming, which is similar to our notion of medium errors has also been studied in this context [7]–[9]. In contrast to the type of fault-tolerant scheduling studied in this paper, however, backoff protocols are not typically static.

Fault tolerance has also been studied in the context of real-time scheduling. There are two kinds of fault-tolerant techniques: the replication approach (also called the primary-backup approach) [10] [11] [12] and the re-execution approach [13] [14]. The re-execution approach is similar in flavor to the idea here where we might transmit the messages again.

VII. Discussion and Conclusions

In this paper, we have provided schedules with sublinear overhead for fault-tolerant transmissions. While the model we consider is simple, the goal of this paper is to provide evidence that sublinear overheads are possible. We argued that these sublinear overhead schedules have the potential to provide benefits in realistic situations and should be investigated further. Many open questions remain. For instance, in practice, one could have a more complex model consisting of tuples $\{(f_1, t_1), (f_2, t_2), ...\}$ model where there are at most f_i medium errors in any consecutive t_i steps and f_i/t_i decreases with t_i. Again, we can, in principle use our method iteratively for this model, but it would be better and less pessimistic to directly generate schedules for this model. Second, real-time systems usually need a more complex message model where messages are generated periodically and have deadlines. We believe sublinear overhead schedules can benefit these tasks, but further work is needed to design these schedules. Finally, the $1/2$-good mapping in this paper is computationally expensive to generate, even though it is short. A constructive method for generating reducible $1/2$-good mapping is highly desirable.

Acknowledgment

This research was supported, in part, by the National Science Foundation (USA) under Grant Numbers CCF-1725647, 1918989, and 2106759.

References

[1] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.

[2] A. Burns, J. Harbin, L. Indrusiak, I. Bate, R. Davis, and D. Griffin, "Air-tight: A resilient wireless communication protocol for mixed-criticality systems," in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2018, pp. 65–75.

[3] K. Agrawal, S. Baruah, and A. Burns, "Fault-tolerant transmission of messages of differing criticalities across a shared communication medium," in *Proceedings of the 27th International Conference on Real-Time Networks and Systems*, 2019, pp. 41–49.

[4] M. A. Bender, J. T. Fineman, and S. Gilbert, "Contention resolution with heterogeneous job sizes," in *European Symposium on Algorithms*. Springer, 2006, pp. 112–123.
[5] P. Raghavan and E. Upfal, "Stochastic contention resolution with short delays," *SIAM Journal on Computing*, vol. 28, no. 2, pp. 709–719, 1998.
[6] M. A. Bender, M. Farach-Colton, S. He, B. C. Kuszmaul, and C. E. Leiserson, "Adversarial contention resolution for simple channels," in *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, 2005, pp. 325–332.
[7] M. A. Bender, J. T. Fineman, S. Gilbert, and M. Young, "Noise-off: A backoff protocol for a dynamic, noisy world," *arXiv preprint arXiv:1402.5207*, 2014.
[8] A. Richa, C. Scheideler, S. Schmid, and J. Zhang, "Competitive throughput in multi-hop wireless networks despite adaptive jamming," *Distributed Computing*, vol. 26, no. 3, pp. 159–171, 2013.
[9] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, 2005, pp. 46–57.
[10] Q. Zheng, B. Veeravalli, and C. Tham, "On the design of fault-tolerant scheduling strategies using primary-backup approach for computational grids with low replication costs," *IEEE Transactions on Computers*, vol. 58, no. 3, pp. 380–393, 2009.
[11] Ching-Chih Han, K. G. Shin, and Jian Wu, "A fault-tolerant scheduling algorithm for real-time periodic tasks with possible software faults," *IEEE Transactions on Computers*, vol. 52, no. 3, pp. 362–372, 2003.
[12] C. Dima, A. Girault, C. Lavarenne, and Y. Sorel, "Off-line real-time fault-tolerant scheduling," in *Proceedings Ninth Euromicro Workshop on Parallel and Distributed Processing*. IEEE, 2001, pp. 410–417.
[13] A. Burns, R. Davis, and S. Punnekkat, "Feasibility analysis of fault-tolerant real-time task sets," in *Proceedings of the Eighth Euromicro Workshop on Real-Time Systems*. IEEE, 1996, pp. 29–33.
[14] J. Huang, J. O. Blech, A. Raabe, C. Buckl, and A. Knoll, "Reliability-aware design optimization for multiprocessor embedded systems," in *2011 14th Euromicro Conference on Digital System Design*. IEEE, 2011, pp. 239–246.