Early Deciding Synchronous Renaming in $O(\log f)$ Rounds or Less

Dan Alistarh[1], Hagit Attiya[2⋆], Rachid Guerraoui[3], and Corentin Travers[4⋆⋆]

[1] EPFL `alistarh@epfl.ch`
[2] Technion `hagit@cs.technion.ac.il`
[3] EPFL `guerraoui@epfl.ch`
[4] Univ. Bordeaux `travers@labri.fr`

Abstract. Renaming is a fundamental problem in distributed computing, in which a set of n processes need to pick unique names from a namespace of limited size. In this paper, we present the first *early-deciding* upper bounds for synchronous renaming, in which the running time adapts to the actual number of failures f in the execution. We show that, surprisingly, renaming can be solved in *constant* time if the number of failures f is limited to $O(\sqrt{n})$, while for general $f \leq n-1$ renaming can always be solved in $O(\log f)$ communication rounds. In the wait-free case, i.e. for $f = n-1$, our upper bounds match the $\Omega(\log n)$ lower bound of Chaudhuri et al. [13].

1 Introduction

Unique names, or identifiers, are a fundamental prerequisite for solving a variety of problems in distributed computing. While in many settings unique names are available, they often come from a very large, practically unbounded namespace, which reduces their usefulness. The *renaming* problem [4], in which a set of processes need to be assigned names from a namespace of small size, is one of the fundamental problems in distributed computing, and a significant amount of work, e.g. [1, 2, 4, 6–10, 17, 18], studied its solvability and complexity in fault-prone distributed systems.

Much of the work on the renaming problem, whether in shared-memory [1, 6] or in message-passing systems [4], has focused on the asynchronous case, in which the processes' steps or messages may be delayed arbitrarily by the scheduler. However, real world systems experience long periods where delays are bounded, and inter-process communication is synchronous, even though processes may still fail by crashing. The complexity of renaming in such a synchronous setting, where processes communicate by message-passing, was investigated by Chaudhuri et al. [13] and Okun [19]. In brief, they found that for n processes, up to $n-1$ of which may fail by crashing, there exist algorithms that assign a tight namespace

⋆ Supported in part by the *Israel Science Foundation* (grant number 1227/10).
⋆⋆ Additional supports from the ANR projects ALADDIN and DISPLEXITY.

of names $1, \ldots, n$ names in $O(\log n)$ rounds of communication. Chaudhuri et al. [13] also showed a matching lower bound.

The analysis in these papers only focused on the case where the *maximum* number of processes, $n-1$, may fail by crashing during an execution. However, the $\Omega(\log n)$ lower bound of [13] does not exclude algorithms that terminate faster when the *actual* number of failures f in the execution is less than $n-1$. Such speculative algorithms, also known as *early deciding*, are known to exist for *consensus* [20] and *set agreement* [12]. Early-deciding protocols achieve consensus in $O(f)$ communication rounds [15], and k-set agreement in $O(\frac{f}{k})$ rounds [16].

It is therefore natural to ask what is the time complexity of early deciding synchronous renaming in a message-passing system. The answer to this question does not appear trivial: for example, a successful strategy for renaming [13] was for each process to obtain a new bit from its name by running a simple one round symmetry-breaking protocol–after $\Theta(\log n)$ rounds, each process has a unique name from 1 to n. However, it is hard to speed up this approach to obtain more bits for the process's name in rounds where there are few failures, without breaking name uniqueness. Another approach [4], where each process proposes a name in each round based on what every other process proposed in previous rounds, until there are no collisions, turns out to be very difficult to analyze when the adversary has a limited budget of f failures.

In this paper, we overcome these challenges, and present the first early-deciding upper bounds for renaming. In short, we find that the complexity of the problem is strongly coupled with the relation between n, the number of processes and f, the failure budget of the adversary. We show that there exists an algorithm that ensures a *tight* namespace of n names, and terminates in a *constant* number of rounds in every execution where $f \in O(\sqrt{n})$, and, in $O(\log f)$ rounds, otherwise. The existence of a constant-time renaming algorithm for non-trivial f is surprising, since the early-deciding bounds for consensus [15] and set agreement [16] are *linear* in f irrespective of the relation with the total number of processes n.

Our second result is an algorithm that improves on the constants in the asymptotic notation, terminating in $\log f + 5$ rounds, and assigning names from 1 to $2n$.

The first algorithm is a slight modification of a result by Okun [19]. His protocol is based on a novel connection between synchronous renaming and the approximate agreement problem [14]. In brief, processes assign temporary ranks to each process identifier that they receive, and perform approximate agreement to converge on an approximate rank for each initial process identifier, within a carefully chosen approximation factor. Each process returns this approximate rank, rounded to the nearest integer, as its name: the protocol ensures that, upon termination, the approximate ranks are far enough apart so that no two processes decide on the same rank. Okun [19] showed that this protocol ensures a tight namespace, preserves the order of the initial identifiers, and terminates in $O(\log n)$ rounds in all executions.

Our main contribution is analyzing this protocol for general f, and showing that it terminates in *constant* time if $f \in O(\sqrt{n})$, and in $O(\log f)$ rounds otherwise. Our analysis characterizes the optimal adversarial strategy for arbitrary values of the parameters n and f—we achieve this by carefully bounding the approximation factor of the approximate agreement protocols relative to the number of failures that the adversary expends in each round, showing that this factor goes down fast, and the algorithm terminates very quickly, if the adversary does not fail a significant fraction of the processes in each round.

The second renaming algorithm we present is simpler, and achieves better constants in the asymptotic notation, while relaxing the size of the namespace to $2n$. The protocol is split in two phases: In the first phase, each process identifies a set of names it is interested in, whose size is proportional to the number of failures that the adversary expends in the phase. In the second phase, processes proceed to progressively halve the size of the set of names they are interested in, until each set is a singleton. At this point, each process may adopt the single name in its set. The key technical difficulty is in assigning each process the "right" set of names at the end of the first phase, so that there are always enough names for the set of participants. To ensure this, we need to relax the total size of the namespace that processes are interested in to be of size $2n$. The halving procedure in the second phase is similar to the $O(\log n)$ round algorithm by Chaudhuri et al. [13].

We therefore show that the time complexity of early-deciding renaming is upper bounded by $O(\log f)$ synchronous rounds for general f, and can be constant for non-trivial $f \in O(\sqrt{n})$. Both algorithms are *adaptive*, since they do not know the number of participants n in advance; they also adapt to the number of failures f in the execution.

Roadmap. We present the problem statement and the model in Section 2, and give an overview of related work in Section 3. In Section 4, we present the analysis of the tight renaming algorithm, while the second algorithm can be found in Section 5. We conclude with a discussion of open questions in Section 6.

2 Model and Problem Statement

Model. We consider a standard synchronous message passing system with n processes $p_1, \ldots, p_n$. Initially, processes have unique identifiers from a namespace of unbounded size. Time is divided into *rounds*, and the processes' clocks are synchronized. Each round proceeds as a sequence of send, receive, and process steps, in which processes may send and receive messages, and perform local computation if necessary. We assume that at most $t < n$ processes may fail by crashing. If crashed, a process stops taking further steps in the execution; a process may fail to send any subset of its messages in rounds in which it crashes.

Let f denote the number of failures in the current execution. We focus on *early-deciding* algorithms, that adapt their time complexity to the actual number of failures in the execution, i.e. whose running time is a function of f only.

Renaming. The *renaming* problem [4] requires that each correct process eventually returns a name, and that the names returned should be unique. The size of the resulting namespace should only depend on the parameters n. The *tight* renaming problem requires that the size of the namespace be exactly n; otherwise, we say that the solution is *loose*.

Approximate Agreement. Consider a small real number $\epsilon > 0$. In the ϵ-*approximate agreement* problem [14], each process p_i starts with a proposal v_i, which is a real number. An approximate agreement algorithm must satisfy the following properties: (1) Each correct process eventually decides a value d_i; (2) For any correct processes p_i and p_j, $|d_i - d_j| \leq \epsilon$; (3) For any correct process p_ℓ, there are processes p_i and p_j with initial values v_i and v_j, such that $v_i \leq d_\ell \leq v_j$.

Notation. Throughout this paper, log denotes the logarithm base two. Moreover, we write $\log f$ instead of $\lfloor \log f \rfloor$. To simplify notation, we assume that $f \geq 1$, which allows us to consider running times of $O(\log f)$ rounds.

3 Related Work

The renaming problem was introduced in [4], where the authors also provide a wait-free solution using $(2n - 1)$ names in an asynchronous message-passing system, and show that at least $(n + 1)$ names are required in the wait-free case. This lower bound on the namespace size for the case of asynchronous solutions was improved to $(2n - 2)$ by Herlihy and Shavit [17], and Rajsbaum and Castañeda [11]. (This lower bound holds when n is a power of a prime number.)

Considerable research has analyzed the upper and lower bounds for renaming in asynchronous setting, in particular in shared memory e.g., [1, 2, 6–9, 18]. For a detailed description of these results, we refer the reader to e.g., [1].

On the other hand, there has been relatively little work on the complexity of synchronous renaming. Herlihy et al. [13] considered wait-free renaming in a synchronous message-passing system, identical to the one we consider in this paper. They prove a lower bound of $\Omega(\log n)$ rounds on the time complexity of the problem in runs where $t = n-1$ processes may crash, and provide a matching algorithm that achieves tight renaming with time complexity $\lfloor \log n \rfloor + 3$ in *every* execution. In contrast, our algorithms are *early deciding*, in that they adapt to the *actual* number of failures f in the current execution, deciding in $O(\log f)$ rounds. On the other hand, the lower bound argument of Herlihy et al. [13] applies to our algorithms as well[5], implying a lower bound of $\Omega(\log n)$ rounds in executions where $f = n - 1$.

In [19], Okun presents a synchronous message-passing algorithm for tight renaming algorithm with $O(\log n)$ time complexity. His algorithm is based on a

[5] While the original lower bound of Herlihy et al. [13] applies only to comparison-based algorithms, an argument by Attiya et al. [5] generalizes this bound to a wider class of algorithms, which includes the ones in this paper.

```
procedure $\mathsf{propose}_i(v_i, \epsilon)$ ;
  $d_i \leftarrow v_i$;
  for each round $r \geq 0$ do
    $\mathsf{broadcast}(d_i)$;
    $\delta \leftarrow \max_{\ell \neq j}(|d_\ell - d_j|)$;
    if $\delta \leq \epsilon$ then return $d_i$;
    else $d_i \leftarrow$ arithmetic average of all values $d_j$ received;
```

Fig. 1. The Approximate Agreement algorithm.

novel connection between renaming and approximate agreement. In brief, processes perform approximate agreement on the rank of each initial identifier, until they are certain that no two processes obtain the same rank. In this paper, we extend this technique by providing a new analysis for minor variation of his algorithm, proving that its running time is in fact $O(\log f)$ in executions with f failures, which is asymptotically optimal.

4 A Tight Renaming Algorithm

In this section, we analyze the tight renaming algorithm of Okun [19] and prove that it terminates in $O(\log f)$ rounds, where $f < n$ is the number of processes that the adversary crashes in the current execution. We begin with a short description of the algorithm; a detailed exposition can be found in the original paper [19].

Algorithm Overview. The algorithm is based on a novel connection between renaming and the approximate agreement problem. First, a simple synchronous approximate agreement (AA) algorithm is introduced. Then, the algorithm runs in parallel at most n separate instances of this approximate agreement algorithm, one for *each* process in the system. The goal is to agree on an approximate rank for each process's initial identifier, which will be the value decided by the corresponding approximate agreement algorithm, rounded to the nearest integer. The key ingredient is to run the approximate agreement algorithm for long enough to ensure that the decision values of the AA protocols corresponding to each initial identifier are sufficiently spaced, ensuring that the rank decided for each process is unique.

The Approximate Agreement Algorithm. The algorithm, whose pseudocode appears in Figure 1, proceeds as follows. Each process starts with an initial value v_i and a desired approximation factor ϵ, which bounds the maximum desired skew between decided values. The process maintains an estimate d_i of its decision value, which is updated in every round to the arithmetic average of all values received. Once all the estimates received in a round are within at most ϵ of each other, the process returns its current estimate.

```
procedure rename_i(v_i) ;
    /* Phase one */
    for each round $r = 1, 2, 3$ do
        broadcast($v_i$);
        $C_r \leftarrow$ the number of distinct identifiers received in round $r$;
    $n \leftarrow C_1$;
    $V \leftarrow$ the set of identifiers received in round 3;
    $\epsilon \leftarrow \frac{1}{10C_2}$;
    /* Phase two */
    for each round $r \geq 4$ do
        for every identifier $id \in V$ do
            Participate in $id$'s instance of the approximate agreement algorithm,
            with initial value $C_2 \cdot \mathsf{rank}_V(id)$ , until the algorithm returns;
            /* $\mathsf{rank}_V(id)$ is the rank of id in the set $V$, in increasing order */
    /* Upon completion of all the approximate agreement algorithms */
    $name_i \leftarrow$ final value in $v_i$'s instance of the approximate agreement algorithm,
    rounded to the nearest integer;
    return $name_i$;
```

Fig. 2. The Tight Renaming Algorithm.

The Renaming Algorithm. Each process p_i starts with an initial name v_i. The algorithm, whose pseudocode appears in Figure 2, has two phases.

The first phase contains the first three rounds, in which processes exchange their identifiers, in order to identify the parameter n and the relative ranks of their identifiers.

In the second phase, which starts at round four, based on the information computed so far, each process proposes a rank for each participating process to a separate instance of the approximate agreement algorithm in Figure 1. Notice that all these agreement instances run in parallel; all messages by a process in a round (one for each AA protocol in which it participates) are packaged into a single composite message, which each process sends in each round. The approximation factor for all these agreement instances is $1/(10C_2)$, where C_2 is the number of distinct identifiers the process received in round 2 of the first phase. (This factor is chosen such that no two identifiers may receive the same final rank from the approximate agreement instances when rounded to the nearest integer.) The algorithm terminates when all the approximate agreement algorithms terminate, ensuring the desired approximation factor.

Name uniqueness. We give a brief overview of the mechanism ensuring name uniqueness; a complete analysis can be found in [19]. Recall that, for each initial identifier id, each process p proposes the rank of id multiplied by C_2/C_3 as the initial value in id's instance of the AA protocol. Obviously, the processes' ranks for the same identifier may be distinct (as a consequence of failures in the first phase). However, a key observation is that they will be distinct in a consistent way: if p proposes rank 6 for id α, and rank 7 for id $\beta > \alpha$, then another process q proposing rank 7 for id α will have to propose rank at least 8 for id β. Analyzing the AA protocol, this will imply that, given any process p, its decision values for distinct ids $\alpha < \beta$ will be at distance at least 1 from each other [19, Theorem 3].

This mechanism might still allow the possibility that two distinct processes decide on the same name when rounding their AA decision value to the nearest integer. This is handled by multiplying the initial ranks with C_2/C_3. This ratio is higher than 1 only when a processor observes crashes between the second and third rounds, and the algorithm ensures that for any ids α and β, their corresponding decision values at any two processes p and q are at distance > 1 from each other [19, Lemma 3]. In turn, this implies that no two processes may decide the same name.

Analysis. Our key observation is that the variant of the synchronous approximate agreement algorithm of Okun [19] presented in Figure 1 guarantees the required approximation factor of $1/(10C_2)$ in a constant number of rounds when $f > \sqrt{n}/2$, and $O(\log f)$ rounds, otherwise. In turn, this implies that the renaming algorithm terminates within three additional rounds.

To prove this, we first introduce some notation. Let $\sigma(S)$ be the *diameter* of a set S, i.e. $\max_{a,b \in S}(|a-b|)$.

For any round $r > 0$ in the execution of the approximate agreement algorithm, let U_r be the multiset of distinct values that processes that are active (i.e., send at least one message) in round r have in the beginning of the round.

Let f_r be the number of processes that crash in round r; for convenience, denote $f_0 = 0$. Denote by δ_r the fraction of processes that crash in round r from among the processes that did not crash before round r; that is, $\delta_r = f_r/(n - \sum_{i=0}^{r-1} f_i)$

We first state the following lemma, bounding the diameter of the set U_{r+1} depending on the diameter of U_r and the fraction of processes that crash in round r; its proof follows [19, Lemma 1].

Lemma 1. $\sigma(U_{r+1}) \leq \frac{2\delta_{r+1}}{1-\delta_{r+1}} \cdot \sigma(U_r)$.

The next lemma bounds the number of rounds needed for the approximate agreement algorithm of Figure 1 to achieve a maximum diameter of $\epsilon = 1/(10n)$ for the set of decisions corresponding to each initial value, when starting with proposal sets of diameter $\leq n$. The bound depends on f, the number of failures that the adversary expends in total, and on the number of failures f_i which the adversary expends in each round $i \geq 1$.

Lemma 2. *Consider an execution of the approximate agreement algorithm of Figure 1, starting with an initial set of diameter $\sigma(U_1) \leq n$, in which at most f processes crash. Let R be the number of rounds needed for the algorithm to reach a diameter $\epsilon \leq 1/(10n)$. The following claims hold:*

- *If $f \leq \sqrt{n}/2$, then R is a constant.*
- *If $\sqrt{n}/2 < f \leq n-1$, then $R \leq 5 \log f + 10$.*

Proof. We assume $n \geq 6$; the claim can be checked for $n \leq 5$ by calculation.

The first case is when $f \leq \sqrt{n}/2$. In this case, notice that the fraction of processes that fail in any round of the protocol is at most $\sqrt{n}/n$, i.e. $\delta_r \leq \sqrt{n}/n$, for all $r \geq 1$. In turn, by Lemma 1, the diameter of the set of values for the current instance of approximate agreement is reduced by at least $2\delta_r/(1-\delta_r) \leq 1/(\sqrt{n}-0.5)$ in each round r. Therefore, after the first 10 rounds, the diameter of this set is at most

$$\frac{n}{(\sqrt{n}-0.5)^{10}} \leq \frac{1}{10n}, \text{ for any } n \geq 6.$$

Therefore the maximum diameter the end of round 10 is $\leq 1/(10n)$, as claimed.

In the second case, we assume that $f > \sqrt{n}/2$. First, notice that δ_r, the fraction of active processes that crash in a round r, can be greater than 1/2 in at most $\lfloor \log_2(f+1) \rfloor + 1$ distinct rounds. Therefore, any execution of at least $5\lfloor \log_2(f+1) \rfloor + 10$ rounds contains at least $4\lfloor \log_2(f+1) \rfloor + 9$ rounds r in which $\delta_r < 1/2$. Lemma 1 implies that the diameter of the set U_r at the end of these rounds is at most

$$\frac{n}{2^{4\lfloor \log_2(f+1) \rfloor + 9}} \leq \frac{n}{2^9 \cdot (f+1)^4} \leq \frac{1}{10n} \text{ for all } n \geq 1,$$

where the last step uses the fact that $f \geq \sqrt{n}/2$. Therefore, in this case, the number of rounds necessary to obtain a maximum diameter of at most $1/(10n)$ is $O(\log f)$. □

We conclude that the resulting renaming algorithm is early deciding, terminating in a constant number of rounds, when $f \leq \sqrt{n}/2$, and $O(\log f)$ rounds, otherwise.

Theorem 1. *For any f, $0 \leq f \leq n-1$, let ℓ_f = constant if $f \leq \sqrt{n}/2$, and $\ell_f = 5\log(f+1) + 10$, otherwise. Then the renaming algorithm in Figure 2 is a tight order-preserving renaming algorithm with time complexity $O(\ell_f)$ and message complexity $O(n^2\ell_f)$ in executions where f processes fail by crashing.*

Proof. We focus on early decision, since the other properties follow from Theorem 4 of [19]. First, recall that the initial value v for each approximate agreement algorithm is computed locally by each process as $\mathsf{rank}_V(\alpha) \cdot C_2/C_3$. Since $\mathsf{rank}_V(\alpha)/C_3 \leq 1$, by the definition of C_2, we obtain that all initial values are between 1 and n. This also implies that our desired approximation factor ϵ is at most $1/(10n)$.

Lemma 2 implies that all instances of approximate agreement that the renaming algorithm executes in parallel terminate in $O(\ell_f)$ rounds, with an approximation factor $\epsilon \leq 1/(10n)$. We obtain that the renaming algorithm terminates in $O(\ell_f)$ rounds, as claimed. □

5 A Loose Renaming Algorithm with Improved Round Complexity

In this section, we present a loose renaming algorithm that terminates in $\log f_1+5$ rounds and uses at most $2n$ names, where n is the number of participating processes and f_1 the number of failures that occur in the first round. Our algorithm extends the non-early deciding renaming algorithm by Chaudhuri, Herlihy, and Tuttle [13]. The latter algorithm is tight; its round complexity, however, depends solely on n and not on the number of failures among the participating processes. Specifically, the namespace when n processes participate is $[1, n]$ and the algorithm terminates in $\log n + 2$ rounds. In the following, the algorithm of [13] is called the *CHT-renaming* algorithm.

Algorithm Overview. The pseudo-code of the algorithm appears in Figure 3. It contains two phases.

In the first phase, which consists of the two first rounds (line 1-line 7), each process selects an interval of names in which it wishes to pick its final name. The size of each interval is upper-bounded by $4f_1$, where f_1 is the number of processes that fail during the first round. The second phase (line 8-line 16) consists of a variant of the CHT-renaming algorithm. Processes use this procedure to progressively shrink the interval of names they are interested in, until obtaining an interval of size 1. The algorithm ensures that no two processes obtain the same interval of size 1, i.e. a single name. Thus, each process can decide the unique name in its final interval. Moreover, it guarantees that in each round, processes holding the largest intervals reduce their interval by one half. Therefore our algorithm terminates in $O(\log f_1)$ rounds.

We begin with a brief description of the CHT-renaming algorithm and then explain how each process selects an initial interval of names.

Definitions and notations. Before describing the algorithm in more details, we introduce some notations, extending those in [13]. An interval I of positive integers is *well-formed* if I is of the form $[d2^j + 1, (d + 1)2^j]$ for some positive integers d, j. Note that for every pair I, J of well-formed intervals, $I \cap J \neq \emptyset \implies I \subseteq J \vee J \subseteq I$. Given a set $\mathcal{I}$ of well-formed intervals, we say that interval $I \in \mathcal{I}$ is *maximal* in $\mathcal{I}$ if for every $J \in \mathcal{I}$, either $I \cap J = \emptyset$ or $J \subseteq I$.

The CHT-renaming Algorithm Each process p maintains a well-formed interval of names I, which are the names p is interested in. In each round, process p sends its id and the interval it currently holds I. The intervals intersecting with I that p receives are stored in the set $\mathcal{I}$ (line 10). p also stores the set of ids of the processes that are conflicting with p, that is, processes that hold an interval intersecting with I in the set $\mathcal{P}$ (line 11). If I is maximal in $\mathcal{I}$ (line 12), I is split in half, and p picks the bottom half or the top half of I, denoted $bot(I)$ and $top(I)$ respectively, as its new interval (line 13-line 14). More precisely, the new interval of p is $bot(I)$ if the rank of p's identity in $\mathcal{P}$ is smaller than $\frac{|I|}{2}$. Otherwise, p selects $top(I)$ as its new interval. Finally, if p observes that every interval sent in the round has size 1, it decides the unique name in its interval. Since in each round the size of maximal interval is at least divided by 2, the

```
1  procedure $\mathsf{rename}_i(id_i)$ ;
       /* Round 1 */
2      $\mathsf{broadcast}(id_i)$;
3      $rk \leftarrow$ the rank of $id$ among the id received in round 1;
       /* Round 2 */
4      $\mathsf{broadcast}(id_i, rk)$;
5      let $rk\_max =$ largest rank received;  for  each $j, 1 \leq j \leq rk\_max$ do
       $h_j \leftarrow |\{\langle id', rk' \rangle : rk' = j\}|$ ;
6      $est\_f \leftarrow \max\{(\sum_{j \in I} h_j) - |I| : I \subseteq [1, rk\_max]\}$;
7      let $j = \lceil \log(est\_f) \rceil$ and $d$ such that $d \cdot 2^j + 1 \leq rk \leq (d+1) \cdot 2^j$ ;
8      $I \leftarrow [d \cdot 2^{j+1} + 1, (d+1) \cdot 2^{j+1}]$ ;
       /* Round $3, 4, \ldots$: CHT-renaming protocol [13] */
9      repeat
10         $\mathsf{broadcast}(id, I)$ ;
11         $\mathcal{I} \leftarrow \{I' : \langle id', I' \rangle$ received and $I \cap I' \neq \emptyset\}$ ;
12         $\mathcal{P} \leftarrow \{id' : \langle id', I' \rangle$ received and $I \cap I' \neq \emptyset\}$ ;
13         if  $\forall I' \in \mathcal{I}, I' \subseteq I$ then
14             let $bot(I)$ and $top(I)$ the bottom half and top half of $I$ respectively;
15             if $rank(id, \mathcal{P}) \leq \frac{|I|}{2}$ then $I \leftarrow bot(I)$ else $I \leftarrow top(I)$
16     until $\forall I' \in \mathcal{I}, |I'| = 1$  ;
17     return $name_i$ where $I = [name_i, name_i]$ ;
```

Fig. 3. The Loose Renaming Protocol.

algorithm terminates after $O(\log c)$ rounds, where c is the size of the largest initial maximal interval.

Name uniqueness relies on the following invariant, which is satisfied by the set of intervals $\mathcal{I}[r]$ held by the processes at the beginning of each round r:

Invariant 1. *For every $I \in \mathcal{I}[r]$, if m processes hold an interval $I' \subseteq I$, then $m \leq |I|$. In particular, this means that I is large enough to allow processes with an interval $I' \subseteq I$ to decide distinct names in the interval I.*

In the original CHT-renaming, the initial interval of each process p is of the form $[1, 2^b]$, where 2^b is the least power of 2 larger than or equal to the number of participating processes from which p has received a message in the first round. The invariant above is thus initially true. In our algorithm, initial intervals are selected differently, but the invariant is still satisfied.

Selection of initial intervals. In the first round, process p broadcasts its id, and ranks its id among the ids it receives (line 2). Let rk_p be the rank obtained by p. If, among the ids of the participating processes, the rank of the id of p is i, and f_1 is the number of failures that occur in the first round, then

$$i - f_1 \leq rk_p \leq i. \qquad (1)$$

This is because p may miss at most f_1 messages from processes with id smaller than i. In the second round, p sends its rank together with its id. It then estimates, based on the ranks it receives, the number of failures that occur in the first round. To that end, p evaluates for each interval of names I the difference between the number of processes ranked in I and the size of I (line 4-line 5). The estimate est_f of the number of failures is then the maximum over all differences. More precisely,

$$est\_f = \max_{I \subseteq [1, rk\_max]} \sum_{i \in I} h_i - |I|.$$

where rk_max is the largest rank received by p and h_i is—to the knowledge of p—the number of processes that rank their id i in the first round. The estimation est_f is upper-bounded by f_1, the number f_1 of failures that occur in the first round. To see why, consider an interval $I = [a, b]$. We have:

$$\sum_{i \in I} h_i \leq |\{q : a \leq rk_q \leq b\}| \leq |\{q : a - f_1 \leq rank(id_q, P) \leq b\}| \leq |I| + f_1, \quad (2)$$

where P is the set of participating processes and $rank(id_q, P)$ is the rank of id_q among the ids of the processes in P. The last inequality follows from Equation 1.

Finally, p selects a well-formed interval. This is performed in two steps. First, p chooses two integers d, j such that the well-formed interval $[d2^j + 1, (d+1)2^j]$ contains rk_p, and 2^j is the least power of 2 larger than or equal to est_f. However, by equation (2), at most $2^j + est\_f \leq 2^{j+1}$ processes may have their rank contained in $[d2^j+1, (d+1)2^j]$. Then, in order to satisfy Invariant 1, p selects the interval $I = [d2^{j+1} + 1, (d+1)2^{j+1}]$ as its initial interval. Since a process p' that selects an interval $\subseteq I$ has its rank $rk_{p'}$ contained in $[d2^j + 1, (d+1)2^j]$, at most $2^{j+1} = |I|$ processes select intervals $I' \subseteq I$. Invariant 1 is thus satisfied, which preserves the correctness of the CHT-renaming algorithm. Finally, notice that the size of each initial interval is at most $4f_1$, where f_1 is the number of failures in the first round. Hence the total running time is $O(\log f_1)$ rounds.

The proof of correctness can be found in a companion technical report [3].

6 Discussion

This paper presents the first early-deciding upper bounds for synchronous renaming. We show that, surprisingly, renaming can be solved in *constant* number of rounds if the number of failures f is limited to $O(\sqrt{n})$, while in the general case, renaming can always be solved in $O(\log f)$ communication rounds. In the wait-free case, i.e. for $f = n-1$, this upper bound is matched asymptotically by the $\Omega(\log n)$ lower bound of Chaudhuri et al. [13]. It remains an open question whether this is tight for other values of f.

References

1. Dan Alistarh, James Aspnes, Keren Censor-Hillel, Seth Gilbert, and Morteza Zadimoghaddam. Optimal-time adaptive strong renaming, with applications to count-

ing. In *PODC'11: Proceedings of the 30th annual ACM symposium on Principles of distributed computing*, pages 239–248, 2011.

2. Dan Alistarh, James Aspnes, Seth Gilbert, and Rachid Guerraoui. The complexity of renaming. In *FOCS'11: Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science*, pages 718–727, 2011.
3. Dan Alistarh, Hagit Attiya, Rachid Guerraoui and Corentin Travers. Early deciding synchronous renaming in $O(\log f)$ rounds or less. Technical report, INRIA, 2012. http://hal.inria.fr/hal-00687555/en/.
4. Hagit Attiya, Amotz Bar-Noy, Danny Dolev, David Peleg, and Rudiger Reischuk. Renaming in an asynchronous environment. *Journal of the ACM*, 37(3):524–548, 1990.
5. Hagit Attiya and Taly Djerassi-Shintel. Time bounds for decision problems in the presence of timing uncertainty and failures. *Journal of Parallel and Distributed Computing*, 61(8):1096–1109, 2001.
6. Hagit Attiya and Arie Fouren. Adaptive and efficient algorithms for lattice agreement and renaming. *SIAM Journal on Computing*, 31(2):642–664, 2001.
7. Elizabeth Borowsky and Eli Gafni. Immediate atomic snapshots and fast renaming. In *PODC '93: Proceedings of the twelfth annual ACM symposium on Principles of distributed computing*, pages 41–51, New York, NY, USA, 1993. ACM.
8. Alex Brodsky, Faith Ellen, and Philipp Woelfel. Fully-adaptive algorithms for long-lived renaming. *Distributed Computing*, 24(2):119–134, 2011.
9. James E. Burns and Gary L. Peterson. The ambiguity of choosing. In *PODC '89: Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 145–157, New York, NY, USA, 1989. ACM.
10. Armando Castañeda and Sergio Rajsbaum. New combinatorial topology bounds for renaming: The upper bound. *Journal of the ACM*, 59(1), March 2012.
11. Armando Castañeda and Sergio Rajsbaum. New combinatorial topology bounds for renaming: The lower bound. *Distributed Computing*, 22(5-6):287–301, 2010.
12. Soma Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation*, 105(1):132–158, 1993.
13. Soma Chaudhuri, Maurice Herlihy, and Mark R. Tuttle. Wait-free implementations in message-passing systems. *Theoretical Computer Science*, 220(1):211–245, 1999.
14. Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. Reaching approximate agreement in the presence of faults. *Journal of the ACM*, 33:499–516, May 1986.
15. Danny Dolev, Rudiger Reischuk, and H. Raymond Strong. Early stopping in byzantine agreement. *Journal of the ACM*, 37(4):720–741, October 1990.
16. Eli Gafni, Rachid Guerraoui, and Bastian Pochon. The complexity of early deciding set agreement. *SIAM Journal on Computing*, 40:63–78, January 2011.
17. Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *Journal of the ACM*, 46(2):858–923, 1999.
18. Mark Moir and James H. Anderson. Fast, long-lived renaming (extended abstract). In *WDAG '94: Proceedings of the 8th International Workshop on Distributed Algorithms*, pages 141–155, London, UK, 1994. Springer-Verlag.
19. Michael Okun. Strong order-preserving renaming in the synchronous message passing model. *Theoretical Computer Science*, 411(40-42):3787–3794, 2010.
20. Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.