RECTIFYING CORRUPTED FILES IN DISTRIBUTED FILE SYSTEMS

Sampath Rangarajan and Donald Fussell

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-90-11 May 1990

Rectifying Corrupted Files in Distributed File Systems [*]

Sampath Rangarajan
Donald Fussell
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712

Abstract

In distributed database or file systems, files may be replicated and distributed over a collection of sites connected by a network. If a file becomes corrupted, it can be restored by comparing it with one or more of its copies residing at other sites and then restoring those pages which have been found to be faulty. Page comparisons can be done by generating signatures using an uncorrupted file and comparing them to signatures from the corrupted file at a common site. The overhead of such a strategy is measured by the number of bits in these signatures that must be transmitted among sites. Let a file be divided into n pages, and let f be the maximum number of pages that could be corrupted. We present a probabilistic comparison algorithm which requires $O(f \log n)$ bits to be transmitted to identify the corrupt pages, which improves on previous results on the growth of communicated bits as functions of both n and of f. If both copies compared are corrupt, only twice the number of bits is required as for the previous case. Further, if multiple copies are used for comparison, then the product of the number of copies times the number of bits sent from each of these copies to the comparison site grows as $O(f \log n)$. We provide a lower bound which establishes the optimality of our algorithm to within a constant factor.

[*]This research was supported in part by the Office of Naval Research under Contract N00014-86-K-0597

1 Introduction

Distributed database or file systems allow files to be replicated and copies kept at a number of communicating sites. One goal of replication is to allow the data to be kept consistent in the face of site failures. In case of such a failure, the corrupted copy can be rectified by comparing it with an uncorrupted one and restoring damaged sections. A brute force way to accomplish this involves using the full versions of the corrupted and uncorrupted copies for comparison. This would involve large communication overhead since an entire file would have to be transmitted between sites; this can be prohibitively expensive when large files are involved.

A better strategy for file restoration is to compress the data before transmitting it to a site performing comparisons. Since a primary goal is to isolate the locations in the corrupted file where errors have occurred so that only relatively small portions of the uncorrupted file have to be transmitted for restoration, signatures of the individual pages (or other appropriate pages) of an uncorrupted file copy and the corrupted file can be generated and transmitted to a site for comparison. Signatures which match are assumed to indicate matching, and thus correct, pages, while mismatched signatures indicate a corrupt page needing restoration. The communication cost for this approach can be measured by the number of bits that have to be sent betweeen the sites containing the corrupt and uncorrupted copies. We will henceforth assume that the site containing the corrupt copy is the comparison site. Since the number of bits involved in the transmission of complete uncorrupted pages when mismatches occur depends on the number of errors that have occurred rather than the efficiency of the error identification algorithm, these are not counted in the communication complexity of the algorithm.

Assume that a file is divided into n pages, where the size of a page defines the granularity to which we wish to pinpoint errors. Existing work in this area assumes that two copies of a file (say) F_1 and F_2 exist on different machines, and one of them (say) F_2 becomes corrupted so that up to f different pages of the file can contain errors. Then F_1 is to be compared with F_2 in order to identify the corrupted pages. We call this problem the single uncorrupted copy comparison (SUCC) problem. Previous results for this problem include both deterministic algorithms where all the corrupted pages are correctly identified and probabilistic algorithms where there is a non-zero probability of misdiagnosis of pages. All these methods have been based on sending signatures of the pages generated from the uncorrupted copy of the file to the site with the corrupted copy, so even allegedly deterministic techniques have an inherent probability of error due to *signature aliasing* where two or more different pages map onto the same signature. In [6], an algorithm was presented which correctly identifies up to one faulty page ($f = 1$) with $\log_2 n$ signatures sent from one site to the other. In [1], a technique was presented which identifies up to two faulty pages ($f = 2$) using $\frac{1}{2}\log_2 n(\log_2 n + 1)$ signatures. In both cases, the signatures are assumed to be b bits in length, where the signature aliasing probability is 2^{-b} [10] [6], with b chosen sufficiently large to allow signature aliasing to be ignored. Otherwise the methods are deterministic, and they identify a superset of the corrupted pages as faulty if more than one (two) pages differ. It is not clear whether these methods can be extended to arbitrary f.

In [2], three different probabilistic algorithms were proposed which work for arbitrary values of f. Of these methods, the the most efficient asymptotically [4] requires the transfer of $O(f(\log n + \log(\frac{1}{\delta}))(\log\log n + \log f + \log(\frac{1}{\delta})))$ bits, where δ is the error probability of the algorithm. Thus, the number of bits grows as $O(f \log f)$ with f and as $O(\log n \log\log n)$ with n. In [9] both a deterministic and a probabilistic algorithm were presented for the SUCC problem; the best asymptotic efficiency is achieved by the probabilistic algorithm, which requires the transfer of $O(f^2 \log n)$ signatures and of the same number of bits as the algorithm from [4] cited above.

In this paper, we present a probabilistic solution to the SUCC problem with a lower overhead (of bits transferred from one site to the other) than existing methods. We prove a lower bound for probabilistic techniques for this problem which matches our upper bound to within a constant. Further, we consider the single corrupted copy comparison (SCCC) problem, where both file copies being compared can have up to f pages in error. Our algorithm for this problem requires only twice the number of bits to be transferred for comparisons as for the SUCC problem. Finally, we consider the multiple copy comparison (MCC) problem, where multiple copies (corrupted or uncorrupted) of files are used for comparison. In this case, we measure the overhead as a product of the number of copies and the number of bits sent from each copy. We show that there is a trade-off between the number of copies of files used and the number of bits sent by each of these copies to the comparison site. In all cases other that the SUCC problem, our results are the first solutions presented for these problems.

2 Summary of Results

The results of our paper are summarized below. Note that all the algorithms presented work with asymptotically high probability, that is $1 - O(\frac{1}{n^\delta})$, $\epsilon > 1$, where the value of δ is determined by constants in the algorithm.

2.1 An upper bound for the single uncorrupted copy comparison (SUCC) problem

For this problem, we propose a strategy which has an overhead of $O(f \log n)$ bits to be transmitted from the site with the uncorrupted copy to the site with the corrupted copy. This result is $O(\log f + \log\log n)$ better than the best of the previously published work [4].

2.2 An upper bound for the single corrupted copy comparison (SCCC) problem

For this problem, we use the same strategy used for the SUCC case and show that it requires only twice the number of bits as the SUCC case to be transmitted from the site with the corrupted copy used for comparison to the site with the corrupted copy under diagnosis. Thus, this problem can also be solved with $O(f \log n)$ bit overhead. This problem has not been considered previously.

2.3 A lower bound for the single copy comparison problem

For any probabilistic algorithm for the SUCC case, we show a lower bound which holds for the entire range of variation of f from a constant to $\Omega(n)$. Specifically, we show that for $f = o(n^{\frac{3}{4}})$, the lower bound is $\Omega(f \log n)$. From this, we argue that the lower bound for the SCCC case is also

$\Omega(f \log n)$. Thus, we show that our algorithms for the SUCC case and the SCCC case are optimal to within a constant factor if $f = o(n^{\frac{3}{4}})$. Further, we show that if $f = \Omega(n)$, the lower bound is $\Omega(n)$ and hence the trivial strategy where the whole file is sent from one site to the other is the best possible asymptotically.

2.4 An upper bound for the multiple copy comparison (MCC) problem

We propose a probabilistic strategy for this problem which has an overhead of $O(f \log n)$ measured in terms of the product of the number of copies and the number of bits transmitted from each of these copies to the site with the copy under diagnosis. Thus, we show that if multiple copies are used in the comparison, a trade-off exists between the number of such copies and the bits transmitted from each of these copies. All copies can be corrupted or uncorrupted.

3 The basic strategy

3.1 The single copy comparison case

We will first discuss the strategy used in the single copy comparison (SUCC and SCCC) case. We follow the same strategy as in [4] in forming the signatures of the files which we wish to compare. First, the two sites generate the signatures for all the pages. Each of the signatures is b bits long. Then, they form m sets of pages, $\{S_1, ...S_m\}$ where the sets are formed in such a way that each page has a probability $\frac{1}{f+1}$ of being in any set S_i. Also, the two sites agree on these sets. Next, a signature is generated corresponding to each of these sets where the signature of each set is the combined signature of the signatures of the page in that set. These combined signatures are also b bits long. We assume that the combined signature is formed by EX-ORing the individual signatures. It has been shown in [6] that if b-bit signatures of a set of pages is Exclusive OR-ed to form one single signature of b bits, the aliasing error probability remains approximately 2^{-b}. The site with the compared copy then sends its m signatures to the site with the copy under diagnosis. Here, each of these m signatures are compared with the corresponding signatures generated by the copy under diagnosis. Once the two sets of m signatures are compared, the diagnosis decisions are made using a simple voting strategy based on an *adaptive threshold* as follows.

A *matching signature* exists when signatures generated by both the copies are identical. We count the number of matching signatures of sets to which a page belongs. If this number equals or exceeds a *threshold value* D_t, then we have confidence in the diagnosis and deem that page to be uncorrupted. Otherwise, the page is deemed corrupt. We call D_t the *diagnosis threshold.* Note that although we use the same strategy as in [4] for generating and comparing the signatures, the two strategies differ in that the threshold used in [4] is 1, whereas we use an adaptive threshold based on the maximum number of corrupted pages (f).

3.2 The multiple copy comparison case

In this case, m signatures are generated from the copy being diagnosed and each of the compared copies using the same technique as for the single copy comparison case. Assume that there are R compared copies and that the copy being diagnosed has *random* and *independent* agreements on how to form the m sets with each of the R compared copies (this requirement will become clear later). Now, these R copies send their m signatures to the site with the copy under diagnosis. Here, each of the m signatures from each of the copies is compared with the corresponding signatures generated from the copy under diagnosis. As in the single copy comparison case, for a particular compared copy, we count the matching signatures of sets to which a page belongs. If this number equals or exceeds a diagnostic threshold value D_t, then we have confidence in our diagnosis and deem that page to be uncorrupted with respect to that compared copy. Otherwise, the page is deemed corrupted with respect to that compared copy. As a next step, we define a *match threshold* M_t. If a page is deemed to be uncorrupted due to comparisons with *at least* M_t compared copies, then that page is deemed uncorrupted. Else, it is deemed corrupt.

4 The Algorithms

In this section, we will formally present the algorithms. The main portion of the algorithm (DIAGNOSE PAGES) is common to both single and multiple copy comparisons (MCC). The difference between these two cases lies in the existence of multiple *diagnosis vectors* and also in using a *match threshold* for MCC both of which are explained below. Note that F' denotes the set of pages deemed by the algorithm to be corrupt.

4.1 The single copy comparison case

The same strategy for forming the vote vector is used for both SUCC and SCCC situations. Let us denote the m signatures generated by the compared copy by $\{k_{c,1}, ..., k_{c,m}\}$ and those generated by the copy under diagnosis by $\{k_{d,1}, ..., k_{d,m}\}$ Note that each of the m signatures contains b bits. As a first step, a *diagnosis vector* of length m, $C[1, .., m]$ is formed in such a way that $C[i] = 1$ if $k_{c,i} = k_{d,i}$ else $C[i] = 0$. Essentially, the diagnosis vector gets a 1 for all those elements corresponding to matching signatures. Next, a *Vote Vector* $V[1, ..., n]$ is initialized to zero. Also, define a *diagnosis threshold* D_{th} and set it to $D_{th} = K\ m\frac{1}{2^b(f+1)}$, where $1 < K \leq 2$.

The steps are as follows:

- Form the diagnosis vector.
- Use Algorithm DIAGNOSE PAGES with the above diagnosis vector as the input to perform the diagnosis.

4.2 The multiple copy comparison case

Now, let us denote the signatures generated by the R compared copies by $\{k_{c_1,1}, ..., k_{c_1,m}\}$, $\{k_{c_2,1}, ..., k_{c_2,m}\}$, ..., $\{k_{c_R,1}, ..., k_{c_R,m}\}$. is set to . Then, R *diagnosis vectors* of length m, $C_j[1, .., m]$ for $1 \leq j \leq R$ is formed in such a way that $C_j[i] = 1$ if $k_{c_j,i} = k_{d,i}$ else $C[i] = 0$. Essentially, as before, the diagnosis vectors get a 1 for all those elements corresponding to matching signatures. As before, define a *diagnosis threshold* D_{th} and set it to $D_{th} = K\ m\frac{1}{2^b(f+1)}$, where $1 < K \leq 2$. Set $M_t = \frac{R}{2}$. Also, initialize a count vector $CO[1, ..., n]$ to zero.

The steps of the algorithm is as follows.

For $1 \leq j \leq R$

- Form the diagnosis vector $C_j[]$.
- Initialize the vote vector $v[1, ..., n]$ to zero.
- Use Algorithm DIAGNOSE PAGES with the above diagnosis vector as the input to perform the diagnosis.
- For $1 \leq j \leq n$, if $v[j] = 1$, then $CO[j] = CO[j] + 1$.

For $1 \leq j \leq n$, if $CO[j] \geq M_t$ then *pagej = uncorrupted*, else *pagej = corrupted*.

4.3 The diagnosis algorithm

Once the diagnosis vector is formed, the algorithm below is used to make the diagnosis decision for both single and multiple copy comparisons.

Algorithm DIAGNOSE PAGES

```
$F' = \{\}$
    For Each Page $j$ {
        For Each Set $S_i$ {
            If $(j \in S_i) \wedge (C[i] = 1)$ then $V[j] = V[j] + 1$
        }
}

For Each Page $j$ {
    If $V[j] < D_{th}$ then $F' = F' \cup j$
}
```

/* End of Algorithm. Set F' contains the corrupt pages */

5 Upper Bounds

5.1 The single copy comparison case

We will now show upper bounds for our algorithms for both the SUCC and SCCC cases. With $O(f \log n)$ bits sent from site with the compared copy to the site with the copy under diagnosis, the algorithm diagnoses all pages with an error probability approaching zero as n increases. Let $P_c(n)$ be the probability that all pages are correctly diagnosed by the algorithm for both SUCC and SCCC. Then we have the following result relating the communication complexity of the algorithm to the probability of correct diagnosis.

Theorem 1: If the number of signatures m sent from site 1 to site 2 is $\omega f \log n$ for some constant ω, then $P_c(n) \rightarrow 1$ as $n \rightarrow \infty$.

Parameter ω in the above theorem determines how fast the error probability approaches zero. That is, if the diagnosis error probability is $1 - O(\frac{1}{n^\delta})$, then ω determines δ.

Corollary 1:

For SCCC, it suffices to transmit twice as many bits as SUCC.

The above result follows from the proof and discussion of Theorem 1 which can be found in the appendix.

5.2 The multiple copy comparison case

For the MCC case, we will show that with an overhead measured in terms of the product of the number of copies and the number of bits transmitted from each of these copies of $O(f \log n)$, the algorithm diagnoses all pages with an error probability approaching zero as n increases. With all the parameters as for the single copy case, the complexity of the MCC algorithm is as follows.

Theorem 2: If the product of the number of signatures m sent from each of the sites containing the comapred copies $\times$ the number fo such sites R is $\gamma f \log n$ for some constant γ, then $P_c(n) \rightarrow 1$ as $n \rightarrow \infty$.

As before, Parameter γ in the above theorem determines how fast the error probability approaches zero. That is, if the diagnosis error probability is $1 - O(\frac{1}{n^\delta})$, then γ determines δ.

Corollary 2: For MCC, if some compared copies are corrupted, it will suffice if only those copies transmitted twice the number of bits as uncorrupted copies.

The above result is similar to the one for SCCC when compared to SUCC.

6 Lower Bound

In this section, we will show a lower bound for any probabilistic algorithm for the single copy comparison case. As a special case, we show that for $f = o(n^{\frac{3}{4}})$, the number of bits that have to be sent from one site to the other is $\Omega(f \log n)$, thus showing that the lower bound matches the upper bound of the previous section to within a constant. Similarly, if $f = \Omega(n)$ the lower bound is $\Omega(n)$, which means that the trivial strategy of sending the whole file from one site to the other is the best possible asymptotically.

First, we will show that a bound which holds for any deterministic algorithm which is allowed to produce incorrect results with some probability ϵ. The strategy that we use to prove the lower bound is based on the definition of communication complexity in [11]. Then we will use results from [12] to show that this bound can be extended to apply to randomized algorithms as well. It should be noted here that an argument which applies to deterministic algorithms with no probability of producing incorrect results was used in [9] to show a similar bound.

The idea of communication complexity [11] is as follows. Let $g(i,j)$ be a function of two n-bit integers i and j. Let us assume that the values of i and j are known only to two persons A and B. If the two persons want to cooperatively compute the value of $g(i,j)$, how many bits of information have to be exchanged between the two of them? Two types of randomized communication complexity where a probability ϵ of failure was allowed were defined in [11]. One is *probabilistic two-way complexity (with error ϵ)* denoted by $C_\epsilon(g; 1 \leftrightarrow 2)$ where bits are sent in both directions, and the other is *probabilistic one-way complexity (with error ϵ)* denoted by $C_\epsilon(g; 1 \rightarrow 2)$ where bits are sent only in one direction. Also, in [12] another type of complexity called the *distributional two-way error complexity (with error ϵ)* denoted by$D_\epsilon(g; 1 \leftrightarrow 2)$ was defined. This is the complexity of any deterministic two-way algorithm computing $g(i,j)$ with average error probability not exceeding ϵ. What we want to derive first is the value of $D_\epsilon(g; 1 \leftrightarrow 2)$ for the file comparison problem where the two persons involved are the sites, one with a correct copy of the file and another with the corrupted copy, and g is a function dependent on the two files.

We use the following observations.

Observation 1: The number of ways in which an integer of length n bits can be converted to another integer of the same length where the two integers differ by at most f bits is given by $\sum_{i=1}^{f} \binom{n}{i}$. Let us denote this by $G_{f,n}$.

Observation 2: For $f = o(n^{\frac{3}{4}})$,

$$G_{f,n} = \sum_{i=1}^{f} \binom{n}{i} \geq \binom{n}{f} \approx \frac{n^f e^{(-\frac{f^2}{2n} - \frac{f^3}{6n^2})}}{f!}$$

The above approximation is from [5].

Observation 3: For $f = \Omega(n)$, $G_{f,n} = \Omega(2^n)$.

From now on, assume without loss of generality that a file consists of n bits, at most f of which could be in error, and we need to pinpoint these errors. In terms of the definition above for the communication complexity, i and j are the two files of n bits each. The function g is such that $g(i,j)$ tells us exactly which bits differ between the two files. Let g take values from the set $V = [v_1, ..., v_r]$. Assume that the value of $g(i,j) = 0$ when $i = j$ and it is X when i *cannot* be transformed to j with at most f bit flips. Then from Observation 1 it can be seen that $|V|$ is bounded by $G_{f,n}$. Each of the two files of n bit integers can have a value in the range $[0, ..., 2^{n-1}]$. The matrix $C = (c_{ij})_{i,j=0}^{2^n-1}$, where each $c_{ij} = g(i,j) \in V$ is called the communication matrix, and this matrix determines the communication problem.

Now, let us assume that this matrix C is known to both the sites. As per the discussion above, i is known to site 1 and j is known to site 2 and they have to cooperatively determine c_{ij}. The lower bound proof will be based on calculation of the minimum number of bits that have to be communicated between the two sites in order to reduce the focus to a submatrix of C from which the correct value of $g(i,j)$ can be picked with probability of error no more than some ϵ.

Theorem 3: The *distributional two-way error complexity (with error ϵ)* denoted by $D_\epsilon(g; 1 \leftrightarrow 2)$ for the single copy comparison problem is $\Omega(\log G(f,n))$.

Theorem 3 states that $D_\epsilon(g; 1 \leftrightarrow 2) = \Omega(\log G(f,n))$, which establishes the bound for deterministic algorithms. This bound can be extended to apply to randomized algorithms using the following result from [12].

Observation 4:

$$C_\epsilon(f; 1 \leftrightarrow 2) \geq \frac{1}{2} D_{2\epsilon}(f; 1 \leftrightarrow 2)$$

□

Using Observation 4 and the obvious fact that the one-way communication complexity of the problem is greater than or equal to the two-way communication complexity, it is clear that our bound applies to both deterministic and randomized algorithms which have some probability of producing incorrect results.

Corollary 3: For $f = o(n^{\frac{3}{4}})$, using Observation 2 and Theorem 3, $D_\epsilon(g; 1 \leftrightarrow 2)$ and hence $C_\epsilon(f; 1 \leftrightarrow 2)$ for the single copy comparison problem is $\Omega(f \log n)$.

Corollary 4: For $f = \Omega(n)$, using Observation 3 and Theorem 3, $D_\epsilon(g; 1 \leftrightarrow 2)$ and hence $C_\epsilon(f; 1 \leftrightarrow 2)$ for the single copy comparison problem is $\Omega(n)$.

7 Comments

Given the lower bound and the complexity of our algorithm, we know that the number of bits that have to be sent has to increase linearly as f. Is it possible to increase the granularity to which we need to pinpoint errors and thus be able to have no increase in the number of bits sent when f increases? We have not shown a lower bound for this case. But, if this idea is used in the algorithm we have proposed, it can be seen that this does not help because the decrease in the probability of having an error-free page negates the positive effect due to coarser granularity of identification.

References

[1] D. Barbara, B. Feijoo and H. Garcia-Molina, "Exploiting Symmetries for Low-Cost Comparison of File Copies," Proc. International Conference on Distributed Computing Systems, San Jose, June 1988, pp. 471–479.

[2] D. Barbara and R. J. Lipton, "A Class of Randomized Strategies for Low-Cost Comparison of File Copies," Computer Science Dept. Princeton University, Technical Report, September 1988.

[3] H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations," Annals of Mathematical Statistics, vol.23, pp. 493–507, 1952.

[4] D. Barbara and R. J. Lipton, "A Randomized Technique for Remote File Comparison," Proc. 9th International Conference on Distributed Computing Systems, June 1989, pp. 12–19.

[5] P. Erdos and J. Spencer, "Probabilistic Methods in Combinatorics," Academic Press, 1974.

[6] W. K. Fuchs, K. Wu and J. Abraham, "Low-Cost Comparison and Diagnosis of Large Remotely Located Files," Proc. Fifth Symposium on Reliability in Distributed Software and Database Systems, January 1986, pp. 67–73.

[7] D. Fussell and S. Rangarajan, "Probabilistic Diagnosis of Multiprocessor Systems with Arbitrary Connectivity," Proc. 19th Intl. Symposium on Fault Tolerant Computing, Chicago, Illinois, July 1989.

[8] L. Lovasz, "Communication Complexity: A Survey," Technical Report, Dept. of Computer Sciences, Princeton University, 1989.

[9] T. Madej, "An Application of Group Testing to the File Comparison Problem," Proc. 9th International Conference on Distributed Computing Systems, June 1989, pp. 237–243.

[10] J. Metzner, "A Parity Structure for Large Remotely Located Replicated Data Files, *IEEE Transactions on Computers*, Vol. C-32, No. 8, August 1983.

[11] A. C. Yao, "Some Complexity Questions related to Distributive Computing," Proc. 11th ACM Symposium on the Theory of Computing, pp. 209–213, 1979.

[12] A. C. Yao, "Lower Bounds by Probabilistic Arguments," Proc. 24th IEEE Symposium on the Foundations of Computer Science, pp. 420–428, 1983.

Appendix

The proof of Theorem 1 requires the following corollary [7] to a theorem by Chernoff [3].

Corollary A.1: Let Z be a binomial random variable with parameters n and p. Then

$$P(Z \leq \epsilon np) \leq \exp\left(-(1-\epsilon)^2 np/2\right), 0 < \epsilon \leq 1$$

$$P(Z \geq \epsilon np) \leq \exp\left(-(\epsilon-1)^2 np/3\right), 1 \leq \epsilon \leq 2$$

The following lemmas are required in order to prove the main result.

Lemma A.1: For both SUCC and SCCC, the probability P_1 that a page containing one or more errors belongs to a set with matching signature is given by $\frac{2^{-b}}{(f+1)}$.

Proof: For a page with errors to be in a set with a matching signature, the signature that is formed by that set should alias to a signature formed by a set of all error-free pages. The probability P_1 of this is given by the probability that the page belongs to that set times the aliasing probability and this is $\frac{1}{(f+1)}.2^{-b}$.

□

Lemma A.2: For SUCC, the probability P_2 that an uncorrupted page belongs to a set with matching signature is lower bounded by $\frac{1}{e(f+1)}$.

Proof: For an uncorrupted page to be in a set with a matching signature, that page is part of the set forming the signature (with probability $\frac{1}{(f+1)}$) *and*

- No corrupted page is in the set forming the signature, the probability of which is $(1-\frac{1}{f+1})^f \approx \frac{1}{e}$ for large f, *or*

- There are corrupt pages forming part of the signature, but the signature aliases to the correct signature, the probability of which is $2^{-b}(1 - \frac{1}{e})$.

So, the probability P_2 of an uncorrupted page belonging to a set with a matching signature is given by

$$\frac{1}{(f+1)}\left(\frac{1}{e} + 2^{-b}(1 - \frac{1}{e})\right)$$
$$\geq \frac{1}{e(f+1)}$$

□

Lemma A.3: For SCCC, the probability P_3 that an uncorrupted page belongs to a set with matching signature is lower bounded by $\frac{1}{e^2(f+1)}$.

Proof: For an uncorrupted page to be in a set with a matching signature, that page is part of the set forming the signature (with probability $\frac{1}{(f+1)}$) *and*

- No corrupted page is in the set forming both the signatures being matched, the probability of which is $\left((1 - \frac{1}{f+1})^f\right)^2 \approx \frac{1}{e^2}$ for large f, *or*
- There are corrupt pages forming part of at least one of the two signatures, but the signatures aliase to the correct signature, the probability of which is $\geq \left(2^{-b}\right)^2(1 - \frac{1}{e^2})$.

So, the probability P_3 of an uncorrupted page belonging to a set with a matching signature is

$$\geq \frac{1}{(f+1)}\left(\frac{1}{e^2} + \left(2^{-b}\right)^2(1 - \frac{1}{e^2})\right)$$
$$\geq \frac{1}{e^2(f+1)}$$

□

Any probabilistic diasnosis algorithm can potentially make two types of error. A *pessimistic error* occurs when an uncorrupted page is incorrectly diagnosed as corrupt, and an *optimistic error* occurs when a corrupt page is incorrectly diagnosed as uncorrupted. Let the probability of making a pessimistic error be P_{pes} and the probability of making an optimistic error be P_{opt}. In order to study the asymptotic properties of a probablistic diagnosis algorithm, we need to calculate the probability that none of the above two errors are made on any of the pages. It is helpful to recall that for any integer $n \geq 0$ and for non negative x,y and $x > y$ and $x, y \leq 1$, $n(x-y) \geq x^n - y^n$

Lemma A.4 Given a file with n pages, the probability that all pages are diagnosed correctly is $P_c(n) \geq 1 - fP_{opt} - (n-f)P_{pes}$.

Proof: The probability that at least one of the f corrupted pages is wrongly diagnosed is given by

$$= \sum_{i=1}^{f} \binom{f}{i} P_{opt}^{i}(1-P_{opt})^{f-i}$$
$$= (P_{opt} + (1-P_{opt}))^f - (1-P_{opt})^f$$
$$\leq fP_{opt}$$

Similarly, the probability that at least one of the $(n-f)$ uncorrupted pages is wrongly diagnosed is given by

$$\leq (n-f)P_{pes}$$

So,

$$1 - P_c(n) \leq fP_{opt} + (n-f)P_{pes}$$
$$P_c(n) \geq 1 - fP_{opt} - (n-f)P_{pes}$$

□

Proof sketch for Theorem 1

From Lemmas *A*.1 and *A*.2 and the algorithm description, it is easy to see that the equations for the error probabilities for the SUCC case are

$$P_{pes} = \sum_{i=0}^{D_{th}-1} \binom{m}{i} (P_2)^i (1-P_2)^{m-i}$$

$$P_{opt} = \sum_{i=D_{th}}^{m} \binom{m}{i} (P_1)^i (1-P_1)^{m-i}$$

Let X be a binomial random variable with parameters m and P_2 and let Y be a binomial random variable with parameters m and P_1. Then, using *Corollary A.1*,

$$P_{pes} = P(X < D_{th}) = P(X < c_1 m P_2) \leq \exp\left(-(1-c_1)^2\, \omega f \log n\, P_2\right)$$

where $c_1 = K2^{-b}e$.
For this equation to hold, $0 < c_1 \leq 1$. According to the algorithm, the value of K should be between 1 and 2. Let $K = 1$. Then for $c_1 \leq 1$, $b \geq \log_2 e$. In this case, if $b = 2$, it will suffice.

The above equation for P_{pes} holds for the SCCC also with P_3 substituted for P_2. In this case, $c_1 = K2^{-b}e^2$. Setting $K = 1$ as before, for $c_1 \leq 1$, $b \geq \log_2 e^2$. In this case, if $b = 4$, it will suffice.

$$P_{opt} = P(Y \geq D_{th}) = P(Y \geq c_2 m P_1) \leq \exp\left(-(c_2 - 1)^2\, \omega f \log n\, P_1\right)$$

where $c_2 = K$
For this equation to hold, $1 \leq c_2 \leq 2$ and this is satisfied by setting K to be between 1 and 2 in the algorithm. This equation for P_{opt} holds for SCCC without any change.

From *Lemma A.4*,

$$P_c(n) \geq 1 - fP_{opt} - (n-f)P_{pes}$$

Substituting the (upper bound) values for P_{opt} and P_{pes} and choosing the proper constant ω, it is easy to see that $P_c(N) \to 1$ when $N \to \infty$.

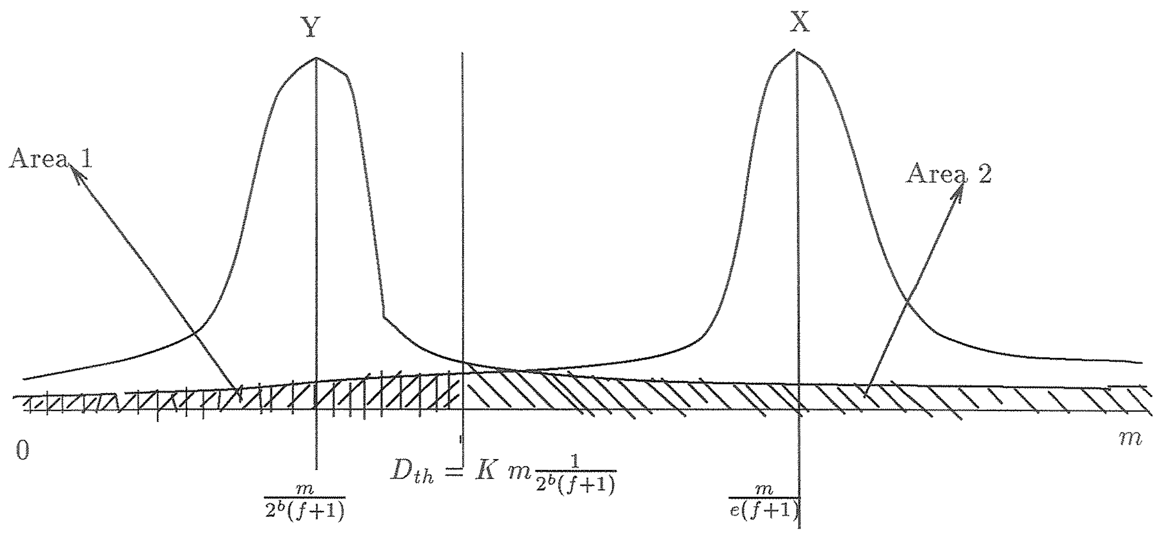

Figure 1: Random Variables representing the number of matching signatures

□

Let us discuss the above theorem. The theorem says that with $\omega f \log n$ signatures each of no more than two bits being sent from one site to the other, the probability of misdiagnosis $\rightarrow 0$ when $n \rightarrow \infty$. That means $O(f \log n)$ bits will suffice. The intuition for our algorithm is as follows. Let us look at SUCC. Similar discussion holds for SCCC. The probability that a page containing one or more errors belongs to a matching signature is given by $\frac{2^{-b}}{(f+1)}$. So, the average number of signatures on which this happens is $m\,\frac{2^{-b}}{(f+1)}$. Similarly, the average number of signatures on which an error-free page belongs to a matching signature is given by $m\,\frac{1}{e(f+1)}$ in case of SUCC (and $m\,\frac{1}{e^2(f+1)}$ in case of SCCC). Random variables Y and X respectively represent the above two cases. *Figure 1* shows the probability density functions for Y and X, and we can see that P_{pes} and P_{opt} of our algorithm are given by areas 1 and 2. We choose our adaptive threshold D_{th} to be between the means of the random variables X and Y. Let us consider the condition on the number of bits per signature b. From the proof sketch, we see that $b \geq \log_2 e$ for SUCC so that $b \geq 2$ and for SCCC $b \geq 2\log_2 e$, so that $b \geq 4$. Thus SCCC requires twice as many bits as SUCC. Thus the algorithm works with $O(f \log n)$ signatures each of 2 bits sent from one site to the other for a total bit complexity of $O(f \log n)$.

The proof of Theorem 2 requires another corollary to a theorem by Chernoff [3].

Corollary A.2: Let Z be a binomial random variable with parameters n and p. Then

$$P(Z \geq \epsilon np) \leq \exp(-\epsilon \log(\epsilon) np), \quad 1 \leq \epsilon$$

Note that this generalized result has a looser condition on ϵ ($1 \leq \epsilon$) compared to the condition on ϵ ($1 \leq \epsilon \leq 2$) found in Corollary A.1.

Proof sketch for Theorem 2

Let $P_{pes}[M]$ and $P_{opt}[M]$ denote the pessimistic and optimistic error probabilities for MCC. The algorithm for MCC makes a pessimitic error iff $\geq M_t$ of the compared copies make pessimistic errors. We have set $M_t = \frac{R}{2}$ in our algorithm. That means, a pessimistic error is made iff $\geq \frac{R}{2}$ compared copies make pessimisstic errors. From the algorithm for MCC, the probability that a single compared copy makes a pessimistic error is P_{pes}, which is the pessimistic error for the single copy case. Similarly, the algorithm for MCC makes an optimistic error if $> \frac{R}{2}$ compared copies make an optmistic error. The probability that a single compared copy makes an optimistic error is P_{opt} which is the optimistic error for the single copy case. For simplicity, we assume that both P_{pes} and P_{opt} could be brought down to less than $\frac{1}{2}$. This condition can be satisfied by sending sufficient bits from each of the compared copies. Note that based on this assumption, we have set M_t to be $\frac{R}{2}$. If this condition is not satisfied, M_t has to be adjusted.

Now, let W be the random variable with parameter R and $p_w = P_{pes}$ and let V be the random variable with parameter R and $p_v = P_{opt}$. Then using *Corollary A.2*,

$$\begin{aligned} P_{pes}[M] = P\left(W \geq \frac{R}{2}\right) &= P\left(W \geq \frac{1}{2P_{pes}} R P_{pes}\right) \\ &\leq \exp\left(-\frac{1}{2P_{pes}} \log\left(\frac{1}{2P_{pes}}\right) R P_{pes}\right) \end{aligned}$$

$$P_{opt}[M] = P\left(V > \frac{R}{2}\right) = P\left(V > \frac{1}{2P_{opt}} K P_{opt}\right)$$
$$\leq \exp\left(-\frac{1}{2P_{opt}} \log\left(\frac{1}{2P_{opt}}\right) R P_{opt}\right)$$

From the proof of *Theorem 1*,

$$P_{pes} \leq \exp\left(-(1-c_1)^2\, m\, P_1\right)$$

and

$$P_{opt} \leq \exp\left(-(c_2-1)^2\, m\, P_2\right)$$

Substituting these values in the equations for P_{pes} and P_{opt}, we get

$$P_{pes}[M] \leq \exp\left(-c_3\, m\, R\, P_1\right)$$

and

$$P_{opt}[M] \leq \exp\left(-c_4\, m\, R\, P_2\right)$$

where c_3 and c_4 are some constants.

From *Lemma A.4*,

$$P_c(n) \geq 1 - f P_{opt} - (n-f) P_{pes}$$

Substituting the values of $P_{opt}[M]$ for P_{opt} and $P_{pes}[M]$ for P_{pes} in the above equation, it is easy to see that with the product $m \times R$ growing as $\gamma \log n$, $P_c(n) \to 1$ as $N \to \infty$, where γ is some constant.

□

Proof sketch for Theorem 3

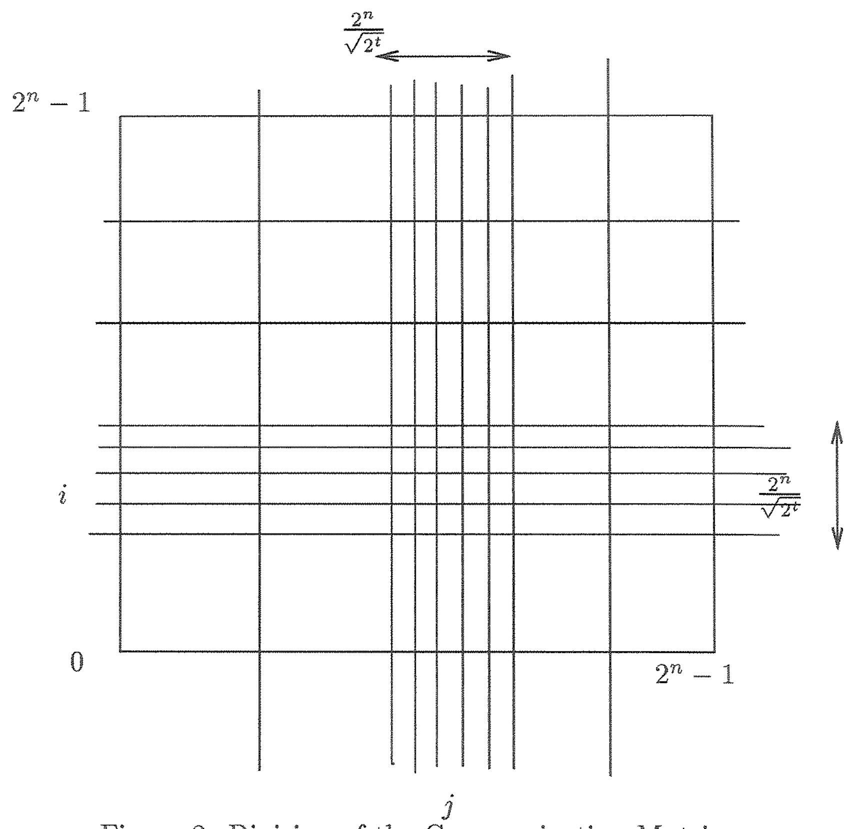

Figure 2: Division of the Communication Matrix

It has been shown in [11] that any Algorithm A computing a function of two variables can be represented in terms of a $4-ary$ decision tree. Essentially every bit transferred from one site to the other takes the algorithm one level down the decision tree. In terms of the communication matrix C, each bit sent from one site to the other will specify which part of the matrix either i or j belongs to. For example, starting at the root with an undivided matrix C, assume that site 1 sends one bit to site 2. Then this bit tells site 2 which of any two sets of rows i belongs to. Thus the matrix has been divided. A bit sent from site 2 to site 1 tells this site which of the two sets of columns j belongs to. In this way, if the division is balanced, by communicating t bits between the two sites, the matrix is divided into 2^t matrices of size $\frac{(2^n)^2}{2^t}$ each. Note that there is a total of $(2^n)^2$ entries in the communication matrix. *Figure 2* shows an example of the above division strategy.

In the best case, if the subdivided matrices are monochromatic, there is no problem picking the correct value of $g(i,j)$. If this is not true, then there is a non-zero probability of making a mistake. Let us now make some observations about the entries of C. It is to be remembered that the entries of the matrix take the value 0 when $i = j$ and the value X when i cannot be transformed to j with at most f bit flips. From the characteristic of the matrix, the number of non 0 or non X entries in a row is bounded by $G_{f,n}$, because an n bit integer can be transformed to another n bit number in at most $G_{f,n}$ ways. Then, the number of non 0 or non X entries in the entire matrix is bounded by $2^n G_{f,n}$. The fraction of the entries in C which are non 0 or non X in the matrix is then $\frac{G_{f,n}}{2^n}$. Consider the scenario after t bits have been communicated and the i and j values have been located to be in a particular submatrix. The size of the submatrix is $\frac{(2^n)^2}{2^t}$. So, the number (say N_1) of non 0 or non X entries in this submatrix is then

$$\frac{G_{f,n}}{2^n}\frac{(2^n)^2}{2^t} = \frac{G_{f,n}2^n}{2^t}$$

Also, consider a specific value v_r from the set V that $g(i,j)$ can take. Again, from the characteristics of the communication matrix, it is clear that the value v_r can occur only once in each row and each column of the matrix. Thus the number of times (N_2) a value v_r can occur in a submatrix is bounded by $\frac{2^n}{\sqrt{2^t}}$. Now, if the correct value of $g(i,j)$ is v_r, the probability with which we pick this value is given by $\frac{N_2}{N_1}$ which is $\frac{\sqrt{2^t}}{G_{f,n}}$. Let this value be greater than $1-\epsilon$ (ϵ is the error probability). Then

$$\frac{\sqrt{2^t}}{G_{f,n}} > 1-\epsilon$$

so that

$$t > c_3 \log G_{f,n}$$

where c_3 is some constant.

Thus, the bit complexity is lower bounded by $\Omega(\log G(f,n))$.

□

For Corollary 3, substituiting for $G(f, n)$ from Observation 2,

$$t > c_3(f \log n - \log f! - \frac{f^2}{2n} - \frac{f^3}{6n^2})$$

Also, $\log f! \leq f \log f$. So,

$$t > c_3(f \log n - f \log f - \frac{f^2}{2n} - \frac{f^3}{6n^2})$$

Using the above equation, for $f = o(n^{\frac{3}{4}})$ it is easy to see that $t = D_\epsilon(g; 1 \leftrightarrow 2) = \Omega(f \log n)$.

Similarly, for Corollary 4, substituiting for $G(f, n)$ from Observation 3, $t = D_\epsilon(g; 1 \leftrightarrow 2) = \Omega(\log G(f, n)) = \Omega(n)$.