Set Reconciliation with Nearly Optimal Communication Complexity[*]

Yaron Minsky,[†] Ari Trachtenberg,[‡] and Richard Zippel[§]

April 29, 2004

Abstract

We consider the problem of efficiently reconciling two similar sets held by different hosts while minimizing the communication complexity, which we call the *set reconciliation* problem. We describe an approach to set reconciliation based on a polynomial encoding of sets. The resulting protocols exhibit tractable computational complexity and nearly optimal communication complexity when the sets being reconciled are sparse. Also, these protocols can be adapted to work over a broadcast channel, allowing many clients to reconcile with one host based on a single broadcast, even if each client is missing a different subset.

1 Introduction

We consider the problem of reconciling two physically separated sets with a minimum of communication. Set reconciliation is particularly useful in systems that make progress in the face of poor and/or unpredictable network connectivity by temporarily sacrificing consistency. Such systems typically

[*]A version of this paper appeared in the *IEEE Transactions on Information Theory*, vol 49, no. 9, pp. 2213-2218, and a different version was presented at the International Symposium on Information Theory, June 2001. A more complete earlier version of this paper is available as Technical Report 2000-1796 from Cornell University

[†]Y. Minsky is with the Dept. of Computer Science, Cornell University, Email: yminsky@cs.cornell.edu

[‡]A. Trachtenberg is with the Dept. of Electrical and Computer Engineering, Boston University, Email: trachten@bu.edu

[§]R. Zippel is with the School of Computer Sciences, The Interdisciplinary Center, Herzliya, Email: rz@idc.ac.il require some mechanism for repairing whatever inconsistencies are introduced, and set reconciliation can be a useful tool for doing those repairs. Examples of this kind of system arise in a variety of contexts, including distributed databases and file systems [1, 2], mobile database synchronization [3, 4], gossip protocols [5–7], and resource location systems [8, 9]. More generally, set reconciliation can be useful in any system that needs to maintain the consistency of unordered, distributed data.

The problem of reconciling two hosts' data sets can be formalized as follows: given a pair of hosts A and B, each with a set of length b bitstrings, how can both host determine the union of the two sets with a minimal amount of communication—both with respect to the number of exchanges between the two hosts and with respect to the number of bits of information exchanged. We call this the *set reconciliation* problem.

This paper presents a surprisingly simple and efficient class of set reconciliation protocols based on a representation of sets as polynomials. The advantage of this approach is that some operations on sets, in particular set difference, can be computed more efficiently from the polynomial encoding. The communication complexity of these set reconciliation protocols is close to the size of the symmetric difference of the two sets. Moreover, under certain conditions, these protocols are one-way [10], meaning that no interaction between hosts is needed. Thus, a host A could broadcast an mb-bit message, and every host B_i whose set differs from A's set by at most m bitstrings (each of length b) could recover the bitstrings it is missing. This works even if each host B_i is missing a different set of bitstrings, so that the total number of distinct bitstrings that can be recovered is much larger than m.

We begin in Section 2 with a brief survey of the relevant literature. Section 3 presents our set reconciliation algorithms and describes their performance. Section 4 presents information-theoretic bounds on set reconciliation, and shows that the communication complexity of our algorithms is near optimal when the sets to be reconciled are sparse. Section 5 discusses some of the connections between our work and Reed-Solomon decoding and Section 6 presents our conclusions and directions for future research. The appendix describes some of the computational intricacies of our protocols.

2 Related work

The general problem of efficiently reconciling similar data is one that has been studied extensively in a number of different settings. An overview of that research is given below.

2.1 Error correcting codes

Set reconciliation is closely related to the problem of error-correction over a noisy channel, where noise corresponds to differences between sets. Thus, one can represent a set $S \subset \mathscr{U}$ by a length $|U|$ bitstring that has a 1 in location i iff the i-th element of U, according to some arbitrary ordering, is present in S. Differences between sets thus correspond to Hamming differences between their bitstring representations, and can be determined using techniques described in Orlitsky's PhD thesis [11] for linear codes, in [12] for Reed-Solomon codes in particular, and in [13] for non-linear codes. Unfortunately, the computational complexity of the standard encoding and decoding algorithms for traditional error-correcting codes generally depends linearly on code size, which is exponential in the representation size of an element of S.

2.2 Communication complexity theory

There is a large body of work in what is typically called "communication complexity theory" in which problems similar to set reconciliation have been studied. The typical scenario for these works is that two discrete random variables with a known joint distribution are given to two independent hosts; each host desires to determine the other's random variable with as little communication as possible. Various results [10, 11, 14, 15] provide information theoretic bounds on the amount of communication needed for such random variable synchronization depending on the amount of interaction (*i.e.*, number of rounds of communication) permitted. These results show that interaction can sometimes substantially reduce the amount of communication needed for synchronization. Extensions of these results show that synchronizing with minimal communication is related to minimal coloring of a corresponding *characteristic graph* [13, 16, 17]. The enormous size of this graph makes direct exploitation of this correspondence unwieldy in practice.

2.3 Set Representation

Central to our approach to set reconciliation in this paper is the representation of sets by their characteristic polynomials. This representation is not novel and was first proposed in [18] and later by Blum and Kannan [19] as part of a probabilistic set equality test. That test is based on a probabilistic equality test for polynomials proposed in [20], and we use essentially the same approach to test rational function equality in Section 3.3. Another aspect of our approach is that polynomials are manipulated by their values rather than their coefficients. A similar approach was proposed by Kaltofen and Trager [21] as a way of efficiently computing with sparse polynomials. They present algorithms for computing the GCD, factorization, and separation of numerators and denominators of black-box rational functions.

Other data structures have been proposed for representing sets in a way that makes various operations more efficient. One relevant example is that of a Bloom filter [22], which is a terse representation of sets that allows for membership queries with a bounded probability of false positives. Unfortunately, Bloom filters do not provide an effective solution to the set reconciliation problem, since the size of a Bloom filter is linear in the size of the set being represented and the probability of false positives can be fairly high.

Another way of encoding a set is to write down the set as a string consisting of the set's elements in sorted order. Insertions and deletions from a set then correspond to insertions and deletions from the corresponding string. Correction of insertion and deletion errors in strings has been studied in the context of spurious error correction [23, 24] and the α-edits problem [15], and a number of algorithms for reconciling such errors have been proposed [12, 25–28]. A great deal of work also been done on the problem of efficiently computing the smallest possible set of edits separating two documents when both documents are available in their entirety (see [29] for an overview.) Correcting insertions and deletion errors in a string appears to be inherently more difficult than correction of insertion and deletion errors in a set because of the ordered nature and unlimited length of the data. In particular, correcting such errors in strings involves determining not only the content of missing/added data, but also its location within a sequence. As a result, set reconciliation algorithms based on this approach incur a logarithmic dependency on the size of the sets being reconciled. Such a dependency does not occur in our algorithms.

3 Set Reconciliation

Consider a pair of hosts A and B that each have a set of length b bitstrings, denoted S_A and S_B respectively. Let the difference sets $\Delta_A = S_A \setminus S_B$ and $\Delta_B = S_B \setminus S_A$ have sizes m_A and m_B respectively. We denote the overall number of differences between the two hosts by $m = m_A + m_B$.

There is one case of the set reconciliation problem that has a straightforward solution—where $m_A = 1$ and $m_B = 0$. In this case, Protocol 1 reconciles the two sets with a single b-bit message. The key to Protocol 1 is that the elements in both S_A and S_B cancel each other out, leaving only the parity sum of the difference set. Since $|\Delta_A| = 1$, this parity sum is in fact the missing bitstring.

Protocol 1 Subset Reconciliation when $m_A = 1$

1. Host A computes parity_A, the parity sum of its bitstrings, and sends it to B
2. Host B computes parity_B, the parity sum of its bitstrings.
3. Host B computes the parity sum of parity_A and parity_B.

3.1 Characteristic Polynomials

Protocol 1 is limited to the case where $m_A = 1$ and $m_B = 0$ by the fact that the parity sum is only sufficient to recover the contents of a singleton set. To generalize the approach of Protocol 1, we need a generalization of the parity sum to the case of multiple differences. The generalization we will use is based on the *characteristic polynomial* $\chi_S(Z)$ of a set $S = \{x_1, x_2, \ldots, x_n\}$, which we define to be the following univariate polynomial.

$$\chi_S(Z) = (Z - x_1)(Z - x_2)(Z - x_3) \cdots (Z - x_n) \tag{1}$$

In order to use characteristic polynomials for set reconciliation, we need to map length b bitstrings onto elements of some field $\mathbb{F}_q$ where $q \geq 2^b$. Note that the zeros of $\chi_S(Z)$ are precisely the elements of S. Thus, the elements of S can be recovered by factoring $\chi_S(Z)$. Also note that $\chi_S(Z)$ is necessarily monic, *i.e.* its leading coefficient is 1.

The characteristic polynomial of a set is not a summary of a set in the same way that the parity sum is, since a set's characteristic polynomial contains all of the information contained in that set. Thus, a host can not transmit the characteristic polynomial of a set any more cheaply than it could transmit the set itself. The characteristic polynomial does, however, allow for the kind of canceling that was central to Protocol 1. In particular, consider the ratio between the characteristic polynomials of S_A and S_B:

$$\frac{\chi_{S_A}(Z)}{\chi_{S_B}(Z)} = \frac{\chi_{S_A \cap S_B}(Z) \cdot \chi_{\Delta_A}(Z)}{\chi_{S_A \cap S_B}(Z) \cdot \chi_{\Delta_B}(Z)} = \frac{\chi_{\Delta_A}(Z)}{\chi_{\Delta_B}(Z)} \tag{2}$$

All terms corresponding to elements that are in both S_A and S_B cancel out, leaving only the characteristic polynomials of Δ_A and Δ_B respectively, from which the difference sets themselves can be recovered.

The problem that remains is, how can A and B efficiently compute the ratio of their characteristic polynomials? The key idea is two divide out the values of the polynomials at a collection of evaluation points, rather than dividing the polynomials directly. The results of these divisions can then be used to interpolate the desired rational function. This approach takes advantage of the fact that if Δ_A and Δ_B are small, then the rational function to be interpolated will have low degree. As a result, the number of evaluation points required will be small as well.

3.2 Reconciliation with a known bound on m.

We first consider how the above ideas can be applied to a context where an upper bound $\overline{m}$ on m is known to all parties. Recall that $m = m_A + m_B$ is the number of elements that differ between S_A and S_B. The case where no bound on m is known is discussed in Section 3.3.

Protocol 2 Set reconciliation

1. Hosts A and B evaluate $\chi_{S_A}(Z)$ and $\chi_{S_B}(Z)$ respectively at the same $\overline{m}$ evaluation points, where $\overline{m}$ is greater than or equal to m.

2. The evaluations of $\chi_{S_A}(Z)$ and $\chi_{S_B}(Z)$ are combined to compute the value of $\chi_{S_A}(Z)/\chi_{S_B}(Z)$ at each of the evaluation points. These values are interpolated to recover the coefficients of the reduced rational function $\chi_{\Delta_A}(Z)/\chi_{\Delta_B}(Z)$.

3. By factoring $\chi_{\Delta_A}(Z)$ and $\chi_{\Delta_B}(Z)$, the elements of Δ_A and Δ_B are recovered.

Protocol 2 outlines our approach to set reconciliation for the case that a close bound $\overline{m}$ on m is known. A more in-depth discussion of the workings of the algorithm is given below.

3.2.1 Evaluating the Characteristic Polynomial

Most of the calculations required for set reconciliation, including the interpolation and factoring, depend only on the size of the symmetric difference between the sets to be reconciled. Evaluating each host's characteristic polynomial at a given evaluation point, on the other hand, requires a linear scan over each host's data set, which may be quite large. If the evaluation points are chosen in advance, however, the cost of the evaluations can be amortized over updates to those data sets. Specifically, adding or deleting an element x involves multiplying or dividing the value of the corresponding characteristic polynomial by $(Z - x)$ for each evaluation point Z. Thus, the cost per insertion or deletion is $2\overline{m}$ field operations.

If an evaluation point k is chosen that is an element of S_A and/or S_B, then the corresponding characteristic polynomial will vanish, complicating the calculation of the ratio of A and B's characteristic polynomials. These anomalous evaluation points can be avoided by increasing the field size q to at least $2^b + \overline{m}$. This ensures there are at least $\overline{m}$ evaluation points that are guaranteed not to coincide with data elements. At worst, this approach require one extra bit per element.

3.2.2 Rational Function Interpolation

The problem of determining a rational function that takes on prescribed values is called the *rational interpolation problem.* In general, given bounds d_1 and d_2 on the degrees of the numerator $P(Z) = \sum_i p_i Z^i$ and denominator $Q(Z) = \sum_i q_i Z^i$ of the rational function to be recovered, and a *support set* V consisting of $d_1 + d_2 + 1$ pairs $(k_i, f_i) \in \mathbb{F}^2$, there is a unique rational function f (up to equivalence) such that $f(k_i) = f_i$ for each $(k_i, f_i) \in V$. Each pair (k_i, f_i) in V implies a linear constraint on the coefficients of the numerator and denominator of the rational function to be recovered:

$$\begin{aligned} k_i^{d_1} + p_{d_1-1} k_i^{d_1-1} + \cdots + p_0 & \\ &= f_i \cdot (k_i^{d_2} + q_{d_2-1} k_i^{d_2-1} + \cdots + q_0). \end{aligned} \quad (3)$$

Interpolation is achieved by solving the $d_1 + d_2 + 1$ simultaneous linear equations implied by the elements of V [30].

Our problem in step 2 of Protocol 2 differs from the standard rational interpolation problem in that we have a bound $\overline{m}$ on the total degree $m = m_A + m_B$ of the function, rather than individual numerator and denominator bounds. Note, however, that $m_A - m_B = |S_A| - |S_B|$, and so can be computed easily. Given $\delta = m_A - m_B$ and an upper bound $\overline{m}$ on m, we can compute bounds on m_A and m_B as follows:

$$m_A \le \lfloor (\overline{m} + \delta)/2 \rfloor \overset{def}{=} \overline{m}_A$$

$$m_B \le \lfloor (\overline{m} - \delta)/2 \rfloor \overset{def}{=} \overline{m}_B$$

We can assume w.l.o.g. that δ and $\overline{m}$ have the same parity[1] so that $\overline{m}_A + \overline{m}_B = \overline{m}$.

The following theorem, adapted from a standard theorem on rational interpolation (see [31], Proposition 2.2.1.4), shows that a support set of size $\overline{m}$ is sufficient to ensure uniqueness.

Theorem 1 *Let V be a support set with $\overline{m}$ elements over a field $\mathbb{F}$. Assume there exist two monic rational functions f and g that satisfy V, and that the numerator and denominator of f (respectively g) have degrees summing to at most $\overline{m}$. If the difference in degrees between numerator and denominator of f is the same as for g, then f and g are equivalent.*

In our case, we know that a rational function f satisfying the degree bounds of Theorem 1 exists because the support set V is taken from the rational function $\chi_{\Delta_A}(Z)/\chi_{\Delta_B}(Z)$, which is assumed to satisfy those degree bounds. Thus, existence and uniqueness (up to equivalence) are guaranteed.

3.2.3 Example

Consider the sets $S_A = \{1, 2, 9, 12, 33\}$ and $S_B = \{1, 2, 9, 10, 12, 28\}$ stored at hosts A and B respectively as 6-bit integers represented by elements of $\mathbb{F}_{97}$. Each host assumes a bound $\overline{m} = 5$ and agrees *a priori* to the evaluation points $E = \{-1, -2, -3, -4, -5\}$.

The characteristic polynomials for A and B are:

$$\begin{aligned}\chi_{S_A}(Z) &= (Z-1)\cdot(Z-2)\cdot(Z-9)\cdot(Z-12)\cdot(Z-33),\\ \chi_{S_B}(Z) &= (Z-1)\cdot(Z-2)\cdot(Z-9)\cdot(Z-10)\cdot\\ &\qquad\qquad\qquad\qquad(Z-12)\cdot(Z-28).\end{aligned}$$

The polynomials are evaluated at the locations in E over $\mathbb{F}_{97}$ to give values:

$Z =$	-1	-2	-3	-4	-5
$\chi_{S_A}(Z)$	58	19	89	77	4
$\chi_{S_B}(Z)$	15	54	68	77	50
$\chi_{S_A}(Z)/\chi_{S_B}(Z)$	75	74	17	1	35

The rational function $\chi_{\Delta_A}(Z)/\chi_{\Delta_B}(Z)$ is recovered by solving the system of equations described in Section 3.2.2. Since the actual symmetric difference

[1]If δ and $\overline{m}$ don't have the same parity, then $\overline{m}-1$ is also a bound on m and trivially has the same parity as δ.

is less then 5, the system of linear equations is singular and we arbitrarily choose the solution

$$\frac{Z^2 + 46Z + 12}{Z^3 + 41Z^2 + 91Z + 4} = \frac{Z - 33}{Z^2 + 59Z + 86}$$

The zeros of the numerator and denominator are $\{33\}$ and $\{10, 28\}$ respectively, which are exactly equal to Δ_A and Δ_B.

3.2.4 Analysis

In order to compute sets Δ_A and Δ_B, Protocol 2 requires $\overline{m}$ evaluations of $\chi_{S_A}(Z)$ and $\chi_{S_B}(Z)$, along with the sizes of S_A and S_B. This leads to a communication complexity of

$$(b+1)\overline{m} + b = (\overline{m}+1)(b+1) - 1 \tag{4}$$

bits. If $\overline{m}$ is chosen near $m = m_A + m_B$, then (4) is close to mb, the cost of simply sending the missing bitstrings. If A recovers the symmetric difference, then an extra $m_A b \leq mb$ bits are required for A to send to B the elements required for B to compute $S_A \cup S_B$. Section 4 compares these results to the information theoretic bounds.

The computational complexity of Protocol 2 has two dominating components: the cost of evaluating the characteristic polynomials $\chi_{S_A}(Z)$ and $\chi_{S_B}(Z)$ at the evaluation points, and the cost of interpolating and factoring. The former component requires $O(|S|\overline{m})$ time, but can be amortized as $O(\overline{m})$ per insertion as noted in Section 3.2.1. The cost of interpolation using Gaussian elimination to solve the system of linear equations is $O(\overline{m}^3)$ operations over $\mathbb{F}_q$, as is the expected cost of simple root finding (see Appendix A). Asymptotically faster algorithms are available, but their practical benefit in this context are unclear.

3.3 Reconciliation without a bound on m

Protocol 2 requires a bound $\overline{m}$ on the size m of the symmetric difference. In the absence of such a bound, a pair of hosts could reconcile their sets by executing Protocol 2 using progressively larger values for $\overline{m}$. Once a sufficiently large $\overline{m}$ is reached, the reconciliation can be completed successfully. Such an approach, however, requires a means for determining whether the value of $\overline{m}$ chosen is large enough.

How can one efficiently test whether the chosen $\overline{m}$ is large enough? One approach would be for hosts A and B to execute Protocol 2 as usual, and check for success at the end. If the interpolation and factorization steps conclude successfully, then the reconciling hosts can test whether the recovered sets are actually equal by, for example, exchanging hashes of those sets. Any such test must be probabilistic in nature in order to be efficient, as we note in Section 4.

It is not necessary to do the complete reconstruction of the sets in order to test whether $\overline{m}$ is large enough. Instead, we can do the necessary test after the interpolation step of the protocol is completed. Let $g(Z)$ be the rational function returned by the interpolation step, and let $f(Z)$ be $\chi_{\Delta_A}(Z)/\chi_{\Delta_B}(Z)$. If $g(Z) = f(Z)$, then $\overline{m}$ must be an upper bound on m. We thus require an efficient means for testing the equality of rational functions.

Rational function equality can be tested probabilistically by evaluating the rational functions in question at a random point and checking whether those values agree. As follows from Theorem 1, the probability of two different monic rational functions agreeing on a randomly selected point is no more than $\rho = (D-1)/|E|$, where E is the subset of F_q from which the evaluation points were chosen and D is a bound on the degrees of the rational functions. In our context, $D = |S_A| + |S_B|$ serves as a trivial upper bound on the degree of the rational functions. Assuming sets S_A and S_B are sparse and $|E|$ is a significant fraction of q, the typical value of ρ will be small and ρ^k will converge to zero quickly in k.

The approach described above requires evaluation points to be chosen at random, and so the evaluation points would need to be sent in addition to the values of the characteristic polynomial at those points. In practice, a pseudo-random number generator can be used to generate evaluation points, thus reducing the amount of data that needs to be sent.

3.3.1 Minimizing bit complexity

If minimizing the number of transmitted bits is a priority, then evaluations could be sent individually. B would simply recompute the interpolated rational function $g(Z)$ every time it receives an evaluation that does not confirm the previous value of $g(Z)$. When B receives k evaluations in a row that confirm the previous value of $g(Z)$, then B accepts $g(Z)$ as equal to $f(Z)$.

The probability that the above protocol terminates with $g(Z) \neq f(Z)$ is bounded above by $m\rho^k$. To achieve a probability of failure less than ϵ, one therefore needs no more than

$$k = \lceil \log_\rho(\epsilon/m) \rceil > \lceil \log_\rho(\epsilon/(|S_A| + |S_B|)) \rceil \tag{5}$$

extra evaluation points. As an example, to achieve a confidence level of 10^{-11} when reconciling sets of 64-bit strings whose combined size is less than $10,000$ would require $k = 1$ extra evaluations.

Using a value of k chosen according to (5) and a pseudo-random number generator for choosing evaluation points would result in the transmission of at most

$$(b + 2)(m + k) + b$$

bits, which is close to the communication complexity of Protocol 2 and the information-theoretic minimum given in Section 4. The communication complexity is close to that given in (4) when $\overline{m}$ is chosen exactly equal to m. The computational complexity of this method is $O(m^4)$ operations, since the interpolation can be repeated as many as m times.

3.3.2 Minimizing round complexity

Sending a single evaluation at a time has the advantage of requiring the computation and transmission of the minimum number of polynomial evaluations. It has the disadvantage, however, of requiring $(m + k)$ rounds of communication. We can reduce the number of rounds to $\lceil \log_c(m + k) \rceil$ by increasing the total number of evaluations by a factor of c each round. Then, in the worst case scenario, the number of extra evaluations sent is $(c - 1)(m + k)$. Equation (5) can be used for selecting k in this case as well.

The communications complexity for this approach is bounded above by

$$(b + 1)c(m + k) + b + \lceil \log_c(m + k) \rceil$$

which is approximately c times the communications complexity of sending one evaluation at a time. Moreover, for fixed c, the computational complexity is $O((m + k)^3)$ operations, which is cubic in m, as opposed to quartic for the one-by-one approach.

4 Information-Theoretic Bounds

The set reconciliation algorithms described in Sections 3 have communications complexity within a small constant of mb. In this section we will show that, for sparse sets, mb is close to the best achievable communication complexity for any set reconciliation protocol.

Let $N = |S_A \cap S_B|$. Set reconciliation demands that host A discern m_B integers from the $2^b - N - m_A$ that it might be missing, and symmetrically for B. This gives the following information-theoretic lower bound on $\hat{C}_\infty$, the number of bits that need to be transmitted between A and B assuming no bound on the number of rounds of communication:

$$\hat{C}_\infty \geq \lg \left[\binom{2^b - N - m_A}{m_B} \cdot \binom{2^b - N - m_B}{m_A} \right]. \tag{6}$$

If $m = m_A + m_B$ is held constant then this expression is minimized when m_A or m_B is zero, as appropriate. This is because, for all $n, j, k \geq 0$ and $j + k \leq n$,

$$\binom{n}{j}\binom{n-j}{k} \geq \binom{n}{j+k},. \tag{7}$$

Thus,

$$\hat{C}_\infty \quad \geq \quad \lg \left[\binom{2^b - N - m}{m} \right]. \tag{8}$$

When 2^b is at least twice as large as either host set, then the lower bound in inequality (8) becomes $(b - 1 - \lg m) \cdot m \approx mb - m \lg m$. Thus,

$$\frac{\hat{C}_\infty}{mb} \geq 1 - \frac{\lg m}{b}.$$

Assuming the sets are sparse, $\lg m$ must be significantly smaller than b, and so $\hat{C}_\infty$ is at best within a small fraction of mb.

The bound in (8) presumes that m is known *a priori*. Without such knowledge, there is no deterministic algorithm for set reconciliation that has better than linear communication complexity in the size of the sets. To see why, note that set reconciliation is strictly more difficult than testing set equality. As follows from a theorem of Yao [32, Theorem 1], the communication complexity of set equality is linear in the size of the sets being tested. It is for this reason that the protocol proposed in Section 3.3 (for the case where no bound on m is known) is probabilistic.

5 Reed-Solomon codes

Set reconciliation can also be accomplished by the use of error-correcting codes, as described in Section 2.1. In fact, the information sent by our algorithm can be understood in terms of the redundancy of a transformed Reed-Solomon code.

Recall that in Protocol 2 we convert a set $S = \{x_1, x_2, x_3, \ldots, x_n\}$ into a characteristic polynomial $\chi_S(Z) = (Z - x_1)(Z - x_2)(Z - x_3) \cdots (Z - x_n)$.

We then evaluate the polynomial at evaluation points p_i to get values

$$\{\chi_S(p_1), \chi_S(p_2), \chi_S(p_3), \ldots \chi_S(p_{\overline{m}})\} \tag{9}$$

for an upper bound $\overline{m}$ on the number of tolerated differences with other sets. These evaluations are transmitted to a reconciling host.

On the other hand, the redundant residue code formulation of Reed-Solomon codes involves converting a message $u = (u_1, u_2, u_3, \ldots, u_k) \in \mathbb{F}_q^k$ into a polynomial $u(Z) = \sum_{i=1}^{k} u_i Z^{i-1}$. In this formulation, the codeword corresponding to the message u is given by

$$c = (u(\alpha^0), u(\alpha^1), u(\alpha^2), \ldots, u(\alpha^{q-2})),$$

where α is a primitive root of unity in $\mathbb{F}_q$. The Chinese Remainder Theorem dictates that $u(\alpha^0) \ldots u(\alpha^k)$ uniquely determine the message u. Thus, the redundant part of the codeword is given by

$$\{u(\alpha^k), u(\alpha^{k+1}), u(\alpha^{k+2}), \ldots u(\alpha^{q-2})\}. \tag{10}$$

If the message u is set to the coefficients of the characteristic polynomial $\chi_S(Z)$, the evaluation points p_i are set to the appropriate powers of α, and k is set to $q - 1 - \overline{m}$, then Equations (9) and (10) are identical. Thus, Protocol 2 transmits the redundancy of a Reed-Solomon encoding of the coefficients of a set's characteristic polynomial.

6 Conclusion

We have examined the problem of reconciling two related sets, stored at separate hosts, with low communication complexity. We have presented and analyzed two protocols for set reconciliation, one deterministic for the case where a bound on the size of the difference is known, and one probabilistic for the case where no such bound is known.

The deterministic protocol requires a tight bound on the number of differences between reconciling hosts, but it does not require interaction and can thus be used in broadcast-style applications. The probabilistic protocol does not require any *a priori* bound on the number of differences between reconciling hosts, but is interactive. The communication complexity of both protocols is within a small constant of the information theoretic lower bound for set reconciliation with sparse sets.

We have implemented all these protocols in a variety of settings [3, 4, 33, 34] and believe that these protocols can serve as a foundation for a new breed of scalable synchronization protocols for distributed applications.

7 Acknowledgments

We are grateful to Ramin Takloo-Bighash, Edward Reingold, Fred Schneider, Lazar Trachtenberg, and Alexander Vardy for stimulating discussions.

References

[1] M. Satyanarayanan, "Coda: A highly available file system for a distributed workstation environment," in *Proceedings of the Second IEEE Workshop on Workstation Operating Systems*, Pacific Grove, CA, September 1989.

[2] A.J. Demers, D. H. Greene, C. Hause, W. Irish, and J. Larson, "Epidemic algorithms for replicated database maintenance," in *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, Vancouver, British Columbia, Canada, August 1987, ACM, number 6, pp. 1–12.

[3] A. Trachtenberg, D. Starobinski, and S. Agarwal, "Fast PDA synchronization using characteristic polynomial interpolation," *Proc. INFOCOM*, June 2002.

[4] D. Starobinski, A. Trachtenberg, and S. Agarwal, "Efficient pda synchronization," *IEEE Trans. on Mobile Computing*, vol. 2, no. 1, January-March 2003.

[5] R.A. Golding, *Weak-Consistency Group Communication and Membership*, Ph.D. thesis, UC Santa Cruz, December 1992, Published as technical report UCSC-CRL-92-52.

[6] R. van Renesse, Y. Minsky, and M. Hayden, "A gossip-style failure detection service," in *Middleware '98: IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, Nigel Davies, Kerry Raymond, and Jochen Seitz, Eds. 1998, pp. 55–70, Springer Verlag.

[7] K. Guo, M. Hayden, R. van Renesse, W. Vogels, and K. P. Birman, "GSGC: An efficient gossip-style garbage collection scheme for scalable reliable multicast," Tech. Rep., Cornell University, December 1997.

[8] R. van Renesse, "Scalable and secure resource location," in *33rd Hawaii International Conference on System Sciences*, January 2000.

[9] M. Harchol-Balter, T. Leighton, and D. Lewin, "Resource discovery in distributed networks," in *18th Annual ACM-SIGACT/SIGOPS Symposium on Principles of Distributed Computing*, Atlanta, GA, May 1999.

[10] A. Orlitsky, "Worst-case interactive communication i: Two messages are almost optimal," *IEEE Trans. on Info. Theory*, vol. 5, no. 36, pp. 1111–1126, 1990.

[11] A. Orlitsky, *Communication Issues in Distributed Computing*, Ph.D. thesis, Stanford University, 1986, Electrical Engineering Department.

[12] K.A.S. Abdel-Ghaffar and A.E. Abbadi, "An optimal strategy for comparing file copies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 1, pp. 87–93, January 1994.

[13] M. Karpovsky, L. Levitin, and A. Trachtenberg, "Data verification and reconciliation with generalized error-control codes," *39th Annual Allerton Conference on Communication, Control, and Computing*, October 2001.

[14] M. Naor, A. Orlitsky, and P. Shor, "Three results on interactive communication," *IEEE Transactions on Information Theory*, vol. 39, no. 5, pp. 1608–1615, September 1993.

[15] A. Orlitsky, "Interactive communication of balanced distributions and correlated files," *SIAM Journal on Discrete Mathematics*, vol. 6, no. 4, pp. 548–564, November 1993.

[16] N. Alon and A. Orlitsky, "Source coding and graphs entropies," *IEEE Transactions on Information Theory*, vol. 42, no. 5, pp. 1329–1339, September 1996.

[17] H.S. Witsenhausen, "The zero-error side information problem and chromatic numbers," *IEEE Trans. on Info. Theory*, vol. 22, no. 5, September 1976.

[18] R. J. Lipton, "Efficient checking of computations," *STACS*, pp. 207–215, 1990.

[19] M. Blum and S. Kannan, "Designing programs that check their work," *Journal of the ACM*, vol. 42, no. 1, pp. 269–291, 1995.

[20] J. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *Journal of the ACM*, vol. 27, pp. 701–717, 1980.

[21] E. Kaltofen and B. M. Trager, “Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators,” *Journal of Symbolic Computation*, vol. 9, no. 3, pp. 300–320 (or 301–320??), 1990.

[22] B. H. Bloom, “Space/time trade-offs in hash coding with allowable errors,” *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[23] V.I. Levenšteĭn, “Binary codes capable of correcting spurious insertions and deletions of ones,” *Problems of Information Transmission*, vol. 1, no. 1, pp. 8–17, 1965.

[24] V.I. Levenšteĭn, “Efficient reconstruction of sequences,” *IEEE Trans. on Info. Theory*, vol. 47, pp. 2–22, January 2001.

[25] D. Barbará and R.J. Lipton, “A class of randomized strategies for low-cost comparison of file copies,” *IEEE Transactions on Parallel Distributed Systems*, pp. 160–170, April 1991.

[26] J.J. Metzner and E.J. Kapturowski, “A general decoding technique applicable to replicated file disagreement location and concatenated code decoding,” *IEEE Transactions on Information Theory*, vol. 36, pp. 911–917, July 1990.

[27] G. Cormode, M. Paterson, S.C. Ṣahinhalp, and U. Vishkin, “Communication complexity of document exchange,” *ACM-SIAM Symposium on Discrete Algorithms*, January 2000.

[28] A. Tridgell, *Efficient algorithms for sorting and synchronization*, Ph.D. thesis, The Australian National University, 2000.

[29] M. Ajtai, R. Burns, R. Fagin, D. D. E. Long, and L. Stockmeyer, “Compactly encoding unstructured inputs with differential compression,” *Journal of the ACM*, vol. 49, no. 3, pp. 289–317, May 2002.

[30] R.E. Zippel, *Effective Polynomial Computation*, Kluwer Academic Press, Boston, 1993.

[31] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 2nd edition, 1993.

[32] A. C. Yao, “Some complexity questions related to distributive computing,” in *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, 1979, pp. 209–213.

[33] Y. Minsky and A. Trachtenberg, "Practical set reconciliation," Tech. Rep., Department of Electrical and Computer Engineering, Boston University, 2002, Technical Report BU-ECE-2002-01.

[34] S. Agarwal, D. Starobinski, and A. Trachtenberg, "On the scalability of data synchronization protocols for PDAs and mobile devices," *IEEE Network*, vol. 16, no. 4, July 2002.

[35] E. Kaltofen and V. Shoup, "Subquadratic-time factoring of polynomials over finite fields," in *27th Annual ACM Symposium on Theory of Computing*, 1995, vol. 9, pp. 398–406.

[36] V. Shoup, "A new polynomial factorization algorithm and its implementation," *Journal of Symbolic Computation*, vol. 20, pp. 363–397, 1995.

A Root Finding of Polynomials

Assume we are given a polynomial $f(Z)$ of degree m over a finite field $\mathbb{F}_q$. This appendix briefly shows how to determine if all the zeros of $f(Z)$ are distinct and lie in $\mathbb{F}_q$ and, if so, how to find them using classical algorithms that require an expected $O(m^3 \lg q)$ field operations. More sophisticated algorithms improve the asymptotic complexity to as low as $O(m^{1.82} \lg q)$ [35], although their basic structure is similar to that presented here and their practical benefits are not clear. The techniques described here are based on well known results (see for instance, [30, 36]) and are included here for completeness.

The particular type of root finding needed by the set reconciliation protocols involves three steps. First, determine if $f(Z)$ is square free. Second, verify that all irreducible factors of $f(Z)$ are linear. And finally, find the linear factors of $f(Z)$.

We can determine if $f(Z)$ is square-free by computing the GCD (greatest common divisor) of $f(Z)$ and its derivative $f'(Z)$ using the Euclidean algorithm in $O(m^2)$ field operations. To verify that $f(Z)$ is the produce of m linear factors, we simply verify that $f(Z) = \mathrm{GCD}(f(Z), Z^q - Z)$, the latter term being the product of all monic linear polynomials over $\mathbb{F}_q$. This verification can be completed in $O(m^2 \log q)$ time by using repeated squaring (mod $f(Z)$), giving an overall verification time of $O(m^2 \log q)$.

Finally, we find the linear factors of $f(Z)$ using probabilistic techniques. We consider two different cases for the field $\mathbb{F}_q$ (corresponding to the possible choices for use in our set-reconciliation protocols): one where q is a prime and the other where $q = 2^b$. When q is a prime, note that the elements of $\mathbb{F}_q$ are zeros of

$$Z^q - Z = (Z^{\frac{q-1}{2}} + 1) \cdot Z \cdot (Z^{\frac{q-1}{2}} - 1).$$

So, almost half of the elements of $\mathbb{F}_q$ are zeros of $R(Z) = Z^{\frac{q-1}{2}} - 1$.

A polynomial with similar properties can also be constructed for the field $\mathbb{F}_{2^b}$:

$$R(Z) = Z^{2^{b-1}} + Z^{2^{b-2}} + \cdots + Z^4 + Z^2 + Z.$$

We then have that

$$\begin{aligned} R(Z) \cdot (R(Z) + 1) &= R(Z)^2 + R(Z), \\ &= Z^{2^b} + Z^{2^{b-1}} + \cdots + Z^2 + R(Z), \\ &= Z^{2^b} + Z. \end{aligned}$$

So, all the elements of $\mathbb{F}_{2^b}$ are zeros of $R(Z) \cdot (R(Z) + 1)$, and each element is either a zero of $R(Z)$ or of $R(Z) + 1$.

To determine the zeros of $f(Z)$, we chose a random element of $a \in \mathbb{F}_q$ and compute GCD(f(Z),R(Z - a)), which will have almost half the degree of $f(Z)$. Applying this technique recursively on the two factors of $f(Z)$, with different values for a will further split the polynomial, ultimately into linear factors. In total, the expected number of GCDs required will be $O(d)$.