Summary - Reconciling Similar Sets

Bowen Song[1]
May 28, 2018

***Abstract*— This report is a part of an independent study for a set and string reconciliation problem in a distributed system. The paper[1] studied for this report is proposing a transmission model for a variant of set reconciliation problem focusing on reducing the communication complexity based on error-correcting codes.**

I. Introduction

The paper is focusing on improving the communication efficiency for set reconciliation with Hamming distance. Like [2] and [3], the protocol is reconciling the symmetric difference between the two sets. The protocol consists of an encoding method to send data through the network between two reconciling hosts with high communication efficiency and a decoding scheme to reconcile data on the other end. The application of the protocol is targeting distributed file system such as the Google File System[4] where every host contains a large body of data and evolves by appending more information without deletion.

The protocol is for a specific kind of set reconciliation situation where elements in the symmetric difference are related through Hamming distance. The situation setup is to have the symmetric difference of the set partitioned into blocks, each within a fixed Hamming distance of each other noted by l. The protocol is then able to function for minimized communication complexity.

II. Algorithm Review

From the protocol setup, there exists a center set consists of all the block centers from each partitioned block. The protocol breaks into two steps, determine the differences and recover the elements. All the information are represented by elements of a *finite field* of characteristic 2. The protocol includes two mappings, $M(\cdot)$ and $f(\cdot)$, that connect set elements with their representations in the finite field. The output codewords have the minimum distance of $2l+1$ between each other with $r = lg(N)$ redundancy. These parameters are determining the effectiveness of the communication. The $M(\cdot)$ mapping is concerning the Hamming distance between elements and the $f(\cdot)$ mapping is an invertible process in which its result from forwarding process on each host gets transferred to the other host and reverse process on the other host for recovering center elements for each block.

[1]B. Song is with Department of Electrical and Computer Engineering, Boston University, Boston MA, sbowen@bu.edu

The key concept is to break a set on each host into blocks and communicate information by embed and compress into two mappings to save communication costs. With information such as block centers, number of blocks, number of elements within each block, and the distance between each element pairs in a block, the symmetric difference can then be reconstructed.

III. Performance Metric

Since this work is based on [5], which is mainly concerning the computation complexity, this paper is only concerning the communication complexity. However, there is no performance metric based on implementation or complexity analysis to follow. A claimed communication complexity is the only performance metric available and is compared to [3] and its former work based on a single block scheme[5].

Expected Complexity	Communication
This Protocol [1]	$O(t^2n + ht^2logn + hltlog(n))$
Protocol of its previous work[5]	$O(n + lhlog(n))$
Protocol from [3]	$O(thn)$

n = code length
t = upper bound on the set symmetric difference
h = maximum number of elements in a block
l = Maximum Hamming distance of each element within a block

TABLE I
PERFORMANCE METRIC

In comparison to the work from its previous protocol[5], we can see a new parameter increasing the complexity by a linear factor. This is because of the complexity introduced by allowing multiple blocks into the algorithm. For comparison to the work from [3], this protocol is conditionally more efficient if t is strictly greater than h and n is significantly large.

Unfortunately, there is no computation analysis available for this protocol which is clearly affected by allowing multiple blocks into the algorithm.

IV. Conclusion

The protocol has its application targeting a common modern distributed file system in which data evolves by mostly appending without deletion. The protocol works well in such a system that is common to have a significantly large amount of data than the symmetric difference between hosts. The partition of blocks is not a recursive process which can help reduce a large amount of computation complexity in comparison to [3].

The protocol is less ideal for facing a less restricted situation in terms of the type of data. There is no adaptation to a one-way protocol. Moreover, since the protocol relies on computing checksums for each block at the time of reconciliation, adopting a suitable data structure to save the checksums could potentially remedy the computation complexity increased by the multiple block scheme.

The protocol presented in this paper is targeting to allow multiple blocks of symmetric difference to reconcile between two hosts based on the work from [5]. The protocol uses the approach of error-correcting codes and reconciles the symmetric difference between sets. The protocol partitions the symmetric differences into blocks to use less complicated error-correcting codes for a minimized communication and computation costs.

References

[1] R. Gabrys and F. F. Hassanzadeh, "Reconciling similar sets," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Oct 2017, pp. 1139–1144.

[2] Y. Minsky, A. Trachtenberg, and R. Zippel, "Set reconciliation with nearly optimal communication complexity," *IEEE Transactions on Information Theory*, vol. 49, no. 9, pp. 2213–2218, Sept 2003.

[3] Y. Minsky and A. Trachtenberg, "Practical set reconciliation," in *40th Annual Allerton Conference on Communication, Control, and Computing*, vol. 248, 2002.

[4] S. Ghemawat, H. Gobioff, and S.-T. Leung, *The Google file system*. ACM, 2003, vol. 37, no. 5.

[5] R. Gabrys and F. Farnoud, "Reconciling similar sets of data," in *Information Theory (ISIT), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 2837–2841.