Summary - Reconciling Graphs and Sets of Sets

Bowen Song[1]
June 11, 2018

Abstract— **This report is a part of an independent study for set and string reconciliation problems in distributed systems. The Paper [1] studied for this report discusses a way of breaking down a set into sets and reconcile them with IBLT [2]. It extends this concept and changes graph reconciliation problem into reconciling each of its vertex signatures.**

I. Introduction

The paper is considering a typical set reconciliation problem where two hosts Set A and Set B are symmetrically different by a relatively small number compared to the number of elements in both sets, and the size of the sets is, therefore, relatively the same. The proposed Sets of Sets Reconciliation (SSR) protocol is very similar to [3] but without the ability to tolerate any small differences. The SSR breaks a set into smaller child sets for reconciliation. Instead of using EMD to measure the similarities of each child set to each child set from another parent set, SSR reconciles the child set with all the child set from the other parent set and look for the successful ones. There are several deviations to this method that would result in different performance.

The second part of the paper introduces the problem of graph reconciliation and provides reconciliation schemes for random graphs and forest. A similar approach to SSR is used in these protocols breaking a graph reconciliation problem into reconciling its vertex signature that relates to the vertex neighborhood.

II. Invertible Bloom Lookup Table

Before we dive into the protocols, we first take a look at the IBLT [2] which is the foundation of these protocols. The IBLT is a data structure with an invertible process for set representation, compression, and difference extraction. For a fixed table size with m cells and k hash functions, each element from a set is inserted into an IBLT as hashed keys in k different locations based on hashed values. Each IBLT cell contains a count of hash assigned keys, an XOR of all inserted keys, and an XOR of checksum of inserted keys. The XOR gives the property that the same element inserted to an IBLT is canceling the previous insertion and, therefore, is considered as the process of removing an element. The inverting process to retrieve inserted elements is based on a peeling process. We find all cells with a count of 1 to recover the elements from their keys. As more keys are removed from the IBLT, more single item cells would be available for peeling process. The peeling process ends when there is no more keys in the table or single item cells.

The IBLT in a set reconciliation protocol starts with Host A inserts all its elements in an IBLT and sends to Host B. Host B inserts all its elements into the received IBLT to obtain the IBLT with the set symmetric difference. Host B then uses the peeling process to retrieve elements in the IBLT till empty or fail by running out of cells with single count item. The failing rate may decrease from increasing IBLT size which is to increase number of cells in an IBLT. Furthermore, an IBLT can accept negative counting. For example, if we set operation insertion from Set A to have a positive counting and negative counting from Set B, during peeling process, we would consider IBLT cells with both -1 and 1 count as part of the symmetric difference that comes from Set B and Set A accordingly. This would save the effort of distinguishing ownership of elements from set symmetric difference.

III. Sets of Sets Reconciliation

The Sets of Sets Reconciliation (SSR) is based on IBLT to reconcile the symmetric difference between sets. The reconciliation process for Sets of Sets requires an upper bound on the number of differences between sets. The protocol uses an estimator data structure to contain the two comparing sets. The estimator provides operations like *update* to add elements to a set, *merge* to union two estimators, and *query* to return an upper bound for the number of symmetric differences. Unfortunately, there is a fail rate of at most δ to retrieve the upper bound which is related to the size of the set. Therefore, the SSR is designed for both knowing (SSRK) and unknowing (SSRU) the upper bound on symmetric differences.

The SSR protocol regards each host with a parent set partitioned into s children sets. Each child set is with at most h elements from a universe of size u. There are several deviations for the protocol design that provides a trade-off between communication and computation complexities. The SSR protocol Theorem 3.3 is merely treating each whole child set as an item from $O(u^h)$ universe. This reduces the protocol into a standard set reconciliation problem based on one IBLT with each item belongs to a very large bitstring. The SSR, however, can sometimes have a substantial

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

number of elements within each child set and end up with a polynomially larger size of universe for all the child sets. This can be improved by representing each child set as length u bit vector as described in Theorem 3.5.

Walking away from the simple model, the SSR protocol Theorem 3.7 treats each child set as an individual set and sets up an IBLT for each child set. The Theorem 3.7 treats each child set with equal $\hat{d}$ as a general upper bound. However, the Theorem 3.7 has the problem of finding child set counterpart from the other host to reconcile with after receiving the IBLTs. A solution to measure set similarities is proposed in [3] by measuring earth mover's distance. The Theorem 3.7 assumes at least one corresponding symmetric different child set in Host B that satisfies the upper bound $\hat{d}$ and takes the approach of attempting to reconcile all $O(\hat{d}^2)$ pairs of child set and keep the successful results. In the case where an upper bound $\hat{d}$ is not available for Theorem 3.7, the SSRU for Theorem 3.7 uses repeated doubling method to guess an upper bound which increases the communication rounds to O(lg(n)).

To reduce the redundancy of reconciliation attempts from all possible pairs of child sets, the SSR protocol Theorem 3.9 creates $t = O(lg(d))$ versions of parent and child IBLTs with increasing child set table size from $O(1)$ to $O(2^t)$. The number of child sets in a parent set, therefore, decreases as child set table size increases. The Theorem 3.9 uses this property to achieve reconciliation of $O(d)$ child sets with $O(1)$ table size to $O(1)$ child sets with $O(d)$ table size. This result in a conditional better performance regarding communication and computation complexities.

The SSR protocol Theorem 3.11 is a hybrid design relying on set difference estimator to define child set differences and select the best protocol for each child set reconciliation. The procedure is as the following:

1) Construct set difference estimator based on child set hashes for each child
2) exchange to estimate set difference for each child
3) identify child set pairs from different hosts with most similarities
4) make a decision* to reconcile

The decision is based on the number of difference between the two child sets. For child set pairs with large differences, the protocol chooses single IBLT reconciliation scheme and for the ones with small differences chooses reconciliation scheme from [4].

Unfortunately, The SSR protocol at this stage is not considering partition child sets recursively and extending the IBLTs to further levels. This Divide-and-Conquer approach could potentially lower the computation complexity.

IV. Graph Reconciliation Problem

In a graph reconciliation setup, Host A and B contains unlabeled graph $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ respectively with $|V_A| = |V_B| = n$. The graph reconciliation problem becomes changing d edges to make G_A and G_B isomorphically equal to each other. The equivalence of one-way scheme is to change G_B to be isomorphic to G_A. For a two-way scheme, G_A and G_B both need to be changed to correspond to the union of the two graphs. Since the graphs are unlabeled, changing graph by adding and deleting edges is not promising unique or isomorphically equal results.

One way to claim two graphs are isomorphically equal is to iteratively compare graph A with all possible isomorphic graphs of B. The proposed method iterating through all possible isomorphic form of a graph is to represent the graph by polynomials $p(\cdot)$ of degree ${}_nC_2$ in a large prime field of size q and increase its lexicographical order. The procedure starts with Host A computes the value of its graph polynomial representation by $p(r)$ where $r \in$ *prime field* and sends the $p_A(r)$ and r value to Host B. Host B computes its graph representation with respect to value r and lexicographically increase the order of its polynomial till $p_B(r) == p_A(r)$. The fail rate of this strategy is $O(n^2/q)$ which correspond to a situation where $p_B(r) == p_A(r)$ and graphs are not isomorphically equal.

V. Random Graph Reconciliation

The Random Graph Reconciliation (RGR) protocol is targeting Erdos-Renyi G(n,p) model random graph which is constructed by connecting nodes randomly with probability p to control its density. For $p > \frac{1}{2}$, the model is more likely to include dense graphs and $p < \frac{1}{2}$ for sparse graphs. The protocol is based on finding a signature for every vertex in a graph which is relabeling invariant. It matches these vertex signatures using isomorphic matching scheme mentioned earlier to determine the equality of the two graphs. The RGR reconciles vertex signatures to obtain a fully reconciled graph. Once the two graphs are reconciled, Host A and B can agree on a labeling and continue the protocol with a normal set reconciliation steps.

A simple approach to RGR is to match each vertex with its counterpart vertex from the other graph via degree sorting, which is termed conforming labeling, and reconcile vertex signature and label graph via Theorem 3.9 and Theorem 3.3 respectively. The RGR introduces a way to measure the difference between two random graphs with a term $(h, a, b) - separated$, where $h =$ *number of vertices with highest degree*, $a =$ *number of maximum degree difference for the highest degree vertices*, and $b =$ *number of maximum degree difference for all other vertices*. The complexity is available in Section VII for reconciling two random graphs that satisfy $(h, d+1, 2d+1) - separated$ with d known as an upper bound for edge differences.

VI. Forrest Reconciliation

The problem of Forrest Reconciliation is defined as reconciling two rooted forests, collections of rooted trees, through inserting or deleting directed edges and result in Forest G_A isomorphic to Forest G_B. This paper recognizes the lack of robust forest signature scheme as a challenge in reconciling forests and, thus, focus on reconciling forests with no large depth. Vertex and edge signatures represent a forest as hashes of its labeling. The paper claims an efficient reconstruction of the forest from vertex and edge signatures. However, this is under assumptions such as unique hashes for signatures as the representation of a forest, and able to group edge signatures based on its identical vertex signatures.

VII. Performance Metric

Complexity	Communication	Computation
IBLT (Known d)	$O(\lceil lg_d(1/\delta)\rceil d\, lg(u))$	$O(\lceil lg_d(1/\delta)\rceil n)$
Basic-Recon (Known d)[4]	$O(d\, lg(u))$	$O(d^3)$
IBLT (Without d)	$O(\lceil lg_d(1/\delta)\rceil d\, lg(u) + lg(1/\delta) lg(n))$	$O(lg(1/\delta)n)$
Theorem 3.3	$O(du\, lg(u)\, lg(n)/lg(d))$	$n\, lg(n)/lg(d)$
Theorem 3.5	$O(du\, lg(n)/lg(d))$	$n\, lg(n)/lg(d)$
Theorem 3.7	$d\, lg(n)(d\, lg(u) + lg(s))/lg(d)$	$(n + d^3) lg(n)/lg(d)$
Theorem 3.9	$d\, lg(n)(lg(d))\, lg(u) + lg(s)$	$(n + d^2) lg(n)\, lg(d)$
Theorem 3.11	$d\, lg(n)(lg(u) + lg(s)/lg(d))$	$(n + d^2) lg(n)$
Random Graph	$O(lg(1/\delta) d(lg(d) lg(h) + lg(n)))$	$O(lg(1/\delta)(\lvert E\rvert + d^2) lg(d))$
Forrest	$O(lg(1/\delta) d\sigma\, lg(d\sigma) lg(n))$	$O(lg(1/\delta)(n + (d\sigma)^2) lg(d\sigma))$

d = symmetric differences between sets
w = $lg(n) + lg(m)$ = number of bits for a word
n = sum of the size of child sets
u = sum of the size of child sets universe
s = number of children set in a parent set
h = number of elements in a child set
δ = $1/poly(n)$ = Probability of success
σ = maximum depth of a tree

TABLE I
Performance Metric

From Table I, as the SSR protocol gets more complicated by the ascending theorem number, the communication complexity increases and the computation complexity decreases. This is the essential trade-off between the proposed SSR protocols. It also relies on a sufficiently small difference between sets value. All proposed protocols are based on the probabilistic IBLT data structure and, therefore, suffer the success rate of $1 - \delta$ which is usually large enough.

The graph reconciliations does have a fairly reasonable communication complexity, however, the computation complexity is not practical. The analysis is based on assumptions without example application or a success rate.

VIII. Conclusion

The Sets of Sets Reconciliation protocol and its extension into graph reconciliation based on IBLT is providing new insight into the field of set reconciliation. However, the paper is lacking experimental result to support the theoretical analysis. While the paper does compare its performance concerning computation complexity to [4], it fails to mention the following work [5] that provides a more competitive result. The idea of SSR, at the current state, is not supporting recursive expansion. A similar protocol that does support such expansion [3] is achieving a better performance. We believe incorporating a similar recursive strategy to expand SSR levels could potentially benefit its performance and can also extend into graph theory for a better result.

The Paper provides the concept of SSR to break down a set into several smaller children sets. The reconciling process uses IBLT for several reasons such as compression and finding the symmetric difference. The protocol can be a one-way scheme and requires one round of communication. It uses an estimator for set symmetric difference to suggest an upper bound for the protocol and bring down the communication and computation complexity. The extension into graph theory uses the same scheme and defines problem as reconciling vertex and edge signatures which can then convert into a set reconciliation problem. The performance may not be as ideal as some of the existing protocols, nevertheless, the scheme setup is not unique to the proposed protocol and can be replaced into a more efficient algorithm.

References

[1] M. Mitzenmacher and T. Morgan, "Reconciling graphs and sets of sets," *arXiv preprint arXiv:1707.05867*, 2017.

[2] M. T. Goodrich and M. Mitzenmacher, "Invertible bloom lookup tables," in *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*. IEEE, 2011, pp. 792–799.

[3] D. Chen, C. Konrad, K. Yi, W. Yu, and Q. Zhang, "Robust set reconciliation," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. ACM, 2014, pp. 135–146.

[4] 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.

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