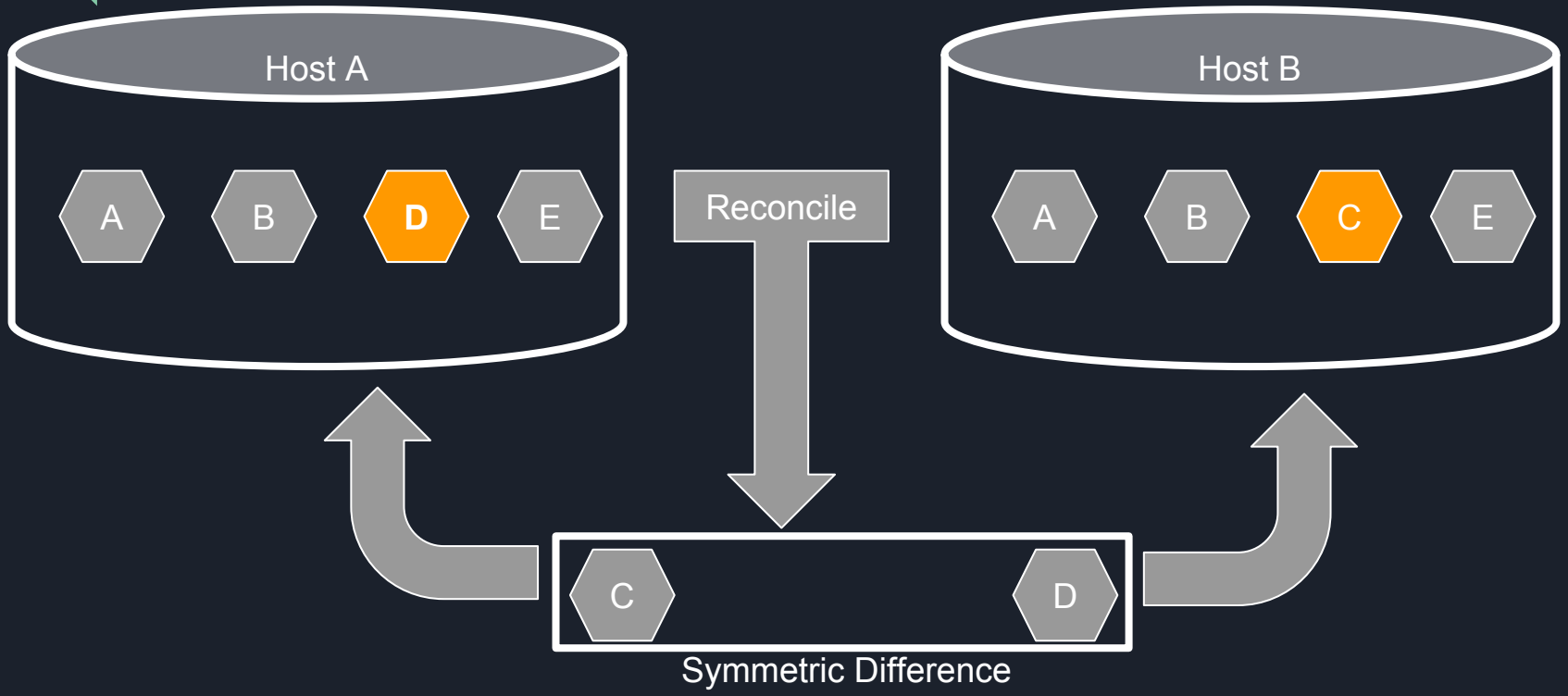
Robust Set Reconciliation[1]

Present by Bowen Song

[1] D. Chen et al, SIGMOD 2014

What is set reconciliation



What is Set Reconciliation used for?

Distributed file system (GFS, Cloud Sync)

Database Synchronization

Example distributed system settings: [2]

- Peer-to-peer sharing
- Partition Healing
- Deduplication
- Synchronizing parallel activations
- Opportunistic ad hoc networks

Presentation Outline

1. Problem Definition
2. Motivation, Challenges, and Contribution
3. Background
    a. EMD - Earth Mover's Distance
    b. IBLT
    c. Quadtree
    d. Randomeshift
4. Algorithm and Demo
5. Analysis and Discussion
6. Experimental Performance
7. Conclusion

1. Problem Definition

- Two or more hosts wish to exchange their data differences for consistency
- Limited communication and computation resources
- Permit incomplete reconciliation

2. Motivation for allowing differences

Differences coming from:

- Noisy transmission channel
- Lossy data compression
- Mathematically equivalent data (rounding errors)
- Privacy-preserving data (intentionally adding controlled noise)

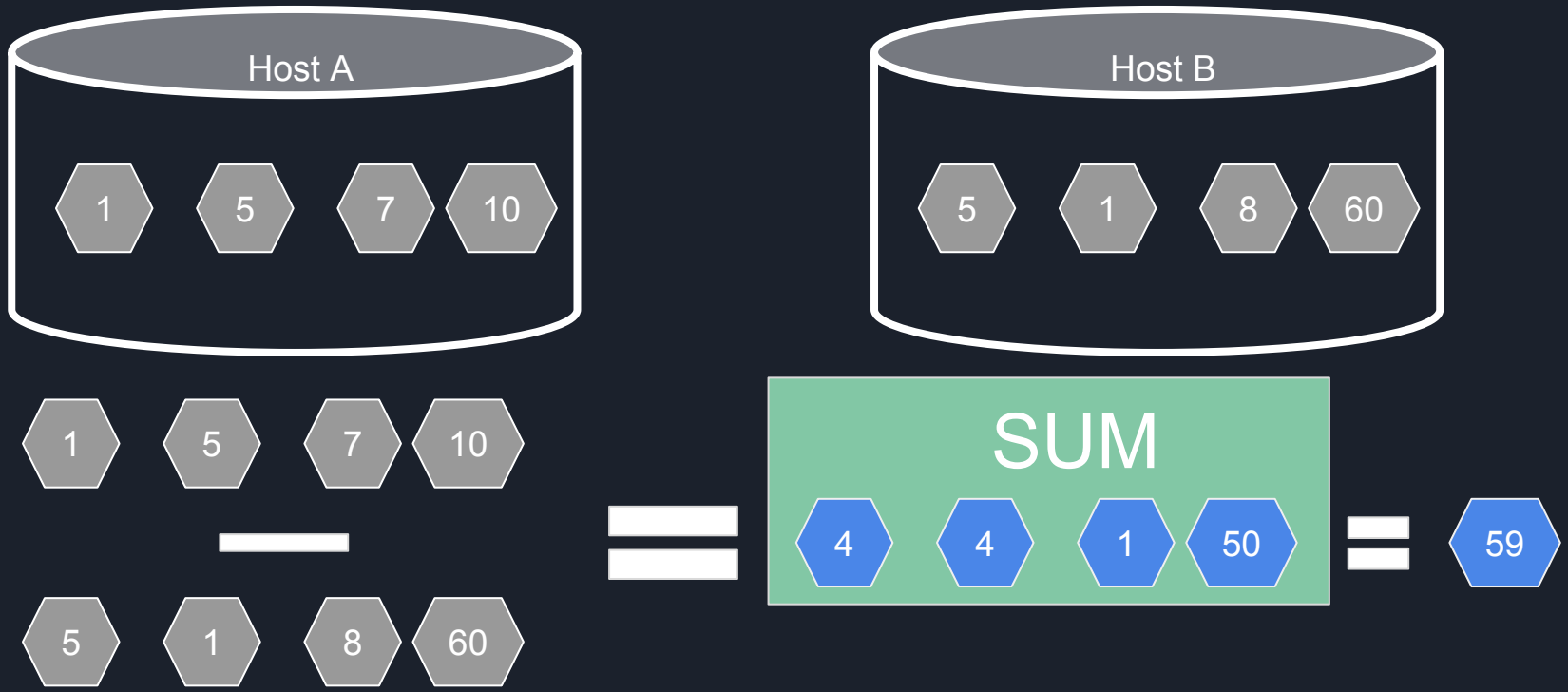
2. Challenges

1. Get the best reconciliation quality:
    a. communication budget
    b. unknown number of differences
2. Efficient in communication and computation
3. Allowing one-way broadcasting reconciliation
4. Measuring differences between sets of (multi-dimensional) data

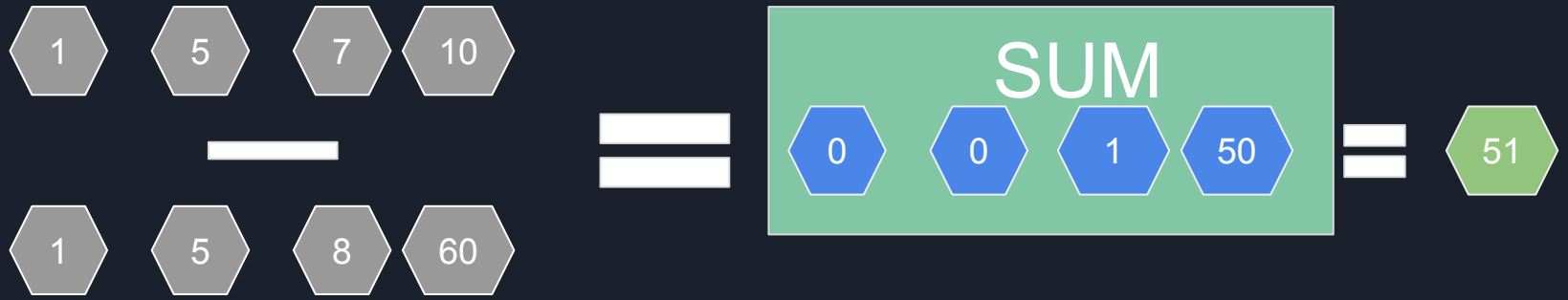
2. Algorithm Contribution

1. Control communication budget trade-off with reconciliation quality
2. Low communication and computation costs
3. One-way and two-way

3a. Earth Mover's Distance (EMD)



3a. Find the smallest bijection



3b. Invertible Bloom Lookup Table (IBLT)

Bloom filter data structure

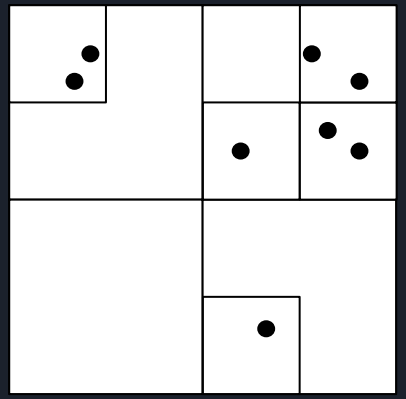
Supporting Operations: **Insert, Delete, List entries**, Subtract[2], Get

Size of Table: allowed number of items for entry listing

Purpose:

- Message Compression
- Extract symmetric differences between two sets

3c. Quadtree



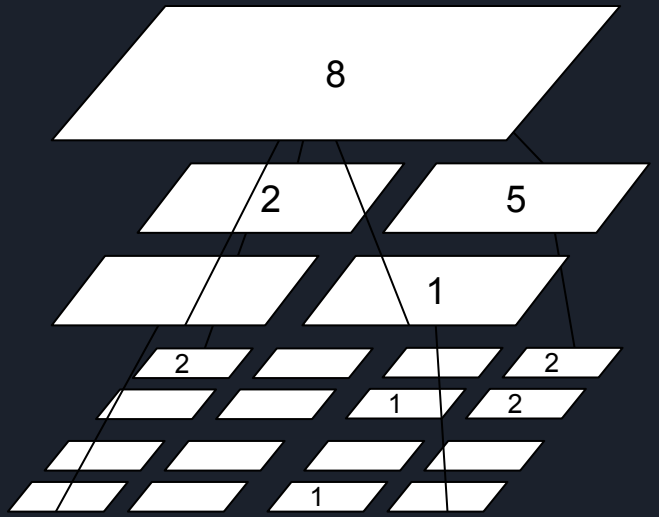
4 children

equal space partition of the parent

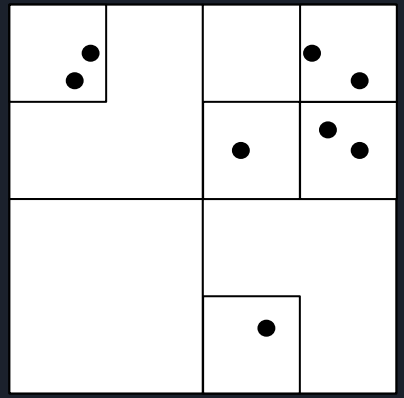
Every node recursively subdivided into four quadrants

Each node:

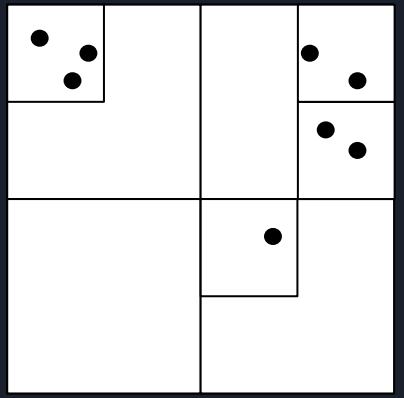
- range of the data points
- Point counts



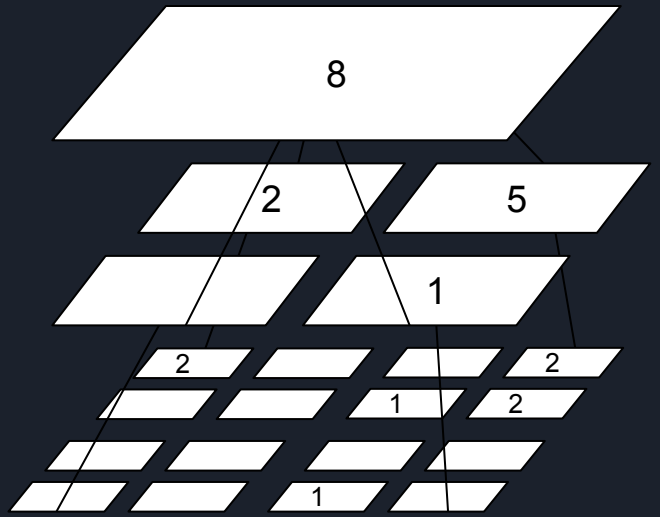
3c. Quadtree

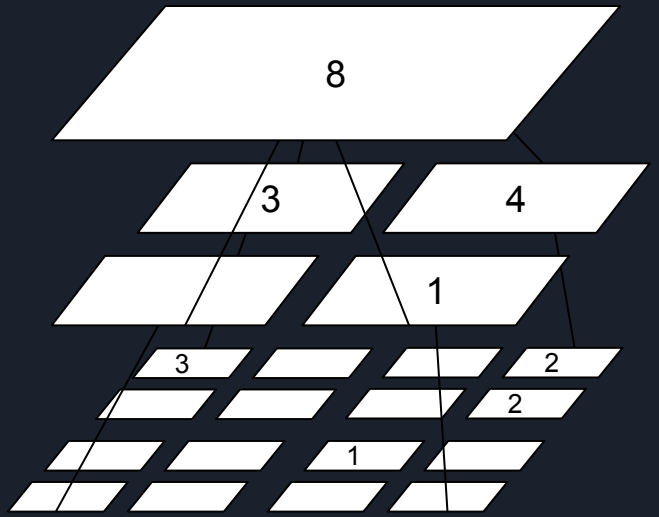


Alice

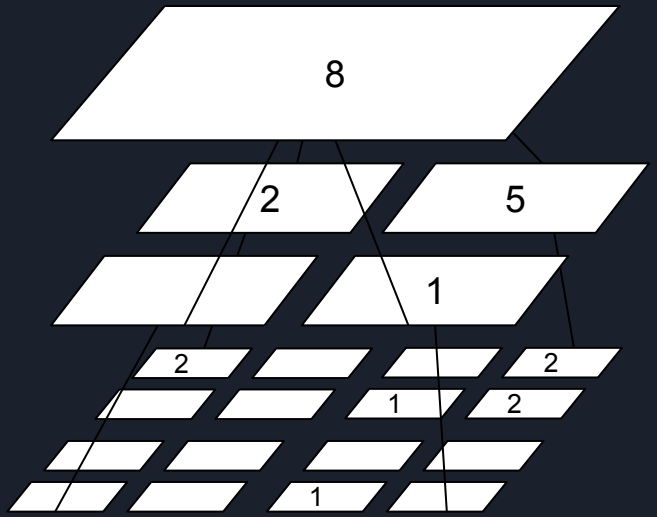


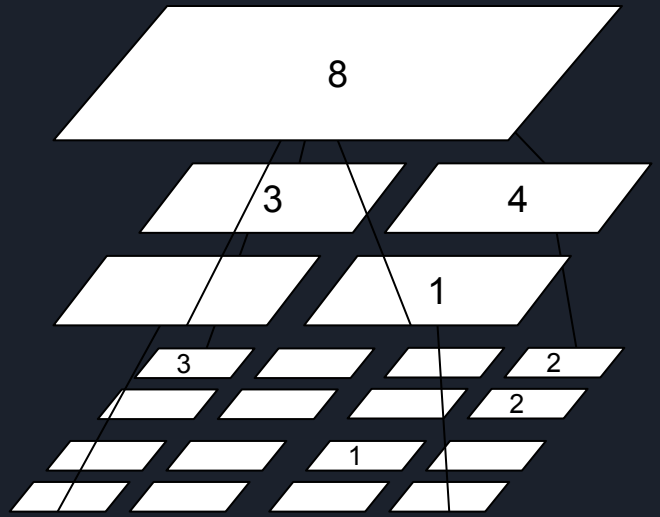
Bob





3c. Quadtree





Differences

0

2

4

3d. Random Shift

Solving: False positive over several reconciliation

Improve the chance of similar item falls into the same grid

Pick a vector $\varepsilon^d = (\varepsilon_1,...,\varepsilon_d)$ uniformly at random from Δ^d

Forward: $S_A' = (S_A + \varepsilon^d) \bmod \Delta^d$

Reverse: $S_A = (S_A' - \varepsilon^d) \bmod \Delta^d$

4. Algorithm Workflow

Alice:

1. Random Shift: Shift all points by ε^d
2. Construct QuadTree: bottom up
3. IBLT: Insert all elements in a layer into a IBLT
4. Send Message: Send ε and the IBLT's

Bob:

1. Random Shift: same ε^d
2. Construct QuadTree
3. Decode IBLT
4. Move/Add points
5. Reverse Random Shift

4. Demo

```
Alice: [ 1.  2.  9. 12. 33.]
Bob: [ 1.  2.  9. 10. 12. 28. 30.]
Int Range: 34.0
Shift By: [2.]
Alice Shifted Set
[ 3.  4. 11. 14.  1.]
Quadtree:
{3.0: 1, 1.0: 1, 11.0: 1, 4.0: 1, 14.0: 1}
{(14, 13): 1, (3, 2): 1, (11, 10): 1, (4, 3): 1, (1, 0): 1}
{(12, 15): 1, (0, 3): 2, (8, 11): 1, (4, 3): 1}
{(0, 15): 5}
{(0, 63): 5}
Insert to IBLTs
Send IBLTs and Eps to Bob
Bob Shifted Set based on Eps
[ 3.  4. 11. 12. 14. 30. 32.]
Quadtree:
{32.0: 1, 3.0: 1, 4.0: 1, 30.0: 1, 11.0: 1, 12.0: 1, 14.0: 1}
{(3, 2): 1, (4, 3): 1, (11, 10): 1, (14, 13): 1, (30, 29): 1, (12, 11): 1, (32, 31): 1}
{(12, 15): 1, (4, 3): 1, (8, 11): 1, (28, 31): 1, (32, 31): 1, (12, 11): 1, (0, 3): 1}
{(32, 31): 1, (16, 31): 1, (0, 15): 5}
{(0, 63): 7}
symmetric difference
('incomplete', □, □)
('complete', [('{(1, 0):1}', '')], [('{(30, 29):1}', ''), ('{(12, 11):1}', ''), ('{(32, 31):1}', '')])
Reverse Random Shift
Reconciled Alice Set
[ 1.   2.   9.  12.  33.  29.5  9.5 27.5]
Reconciled Bob Set
[ 1.   2.   9.  10.  12.  28.  30.  32.5]
```

5. Analysis

Communication Cost: $O(\alpha k \log(n\Delta^d)\log(\Delta))$
Computation Cost: $O(dn \log(\Delta))$

α = Redundancy factor
d = Dimension of an item
Δ = Size of grid including both sets (range)
n = Number of items in a set
k = Fixed number of symmetric differences to be reconciled

5. Discussion

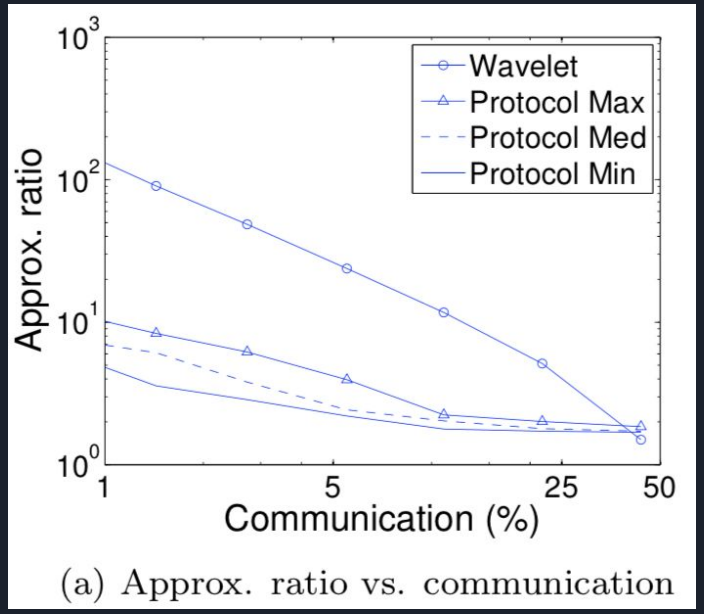
Pros

- Single round message (reduced latency)
- Message is compressed by IBLT
- Trade-off tolerable difference with communication budget
- One-way broadcast or Two-way reconcile

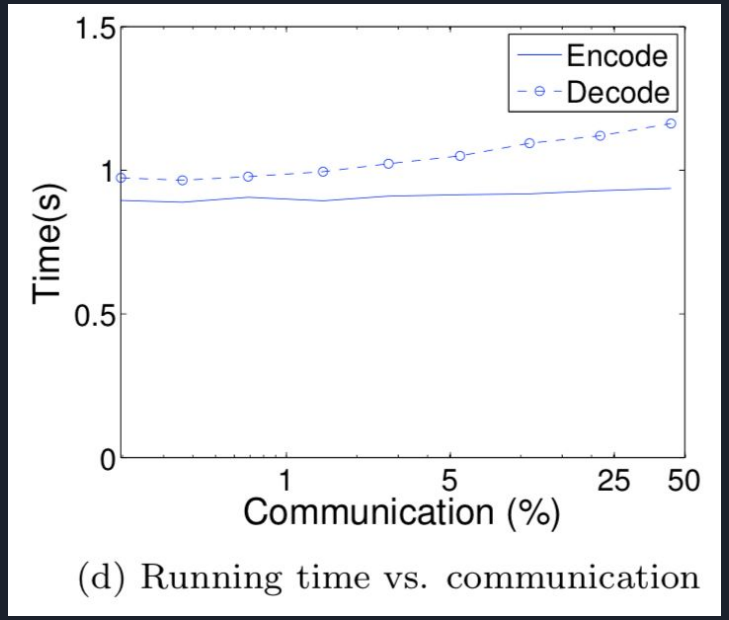
Cons

- Not Resumable
- Probabilistic model
- If leaf level not decodable:
  - False positive
  - Introduce error

6. Experimental Performance

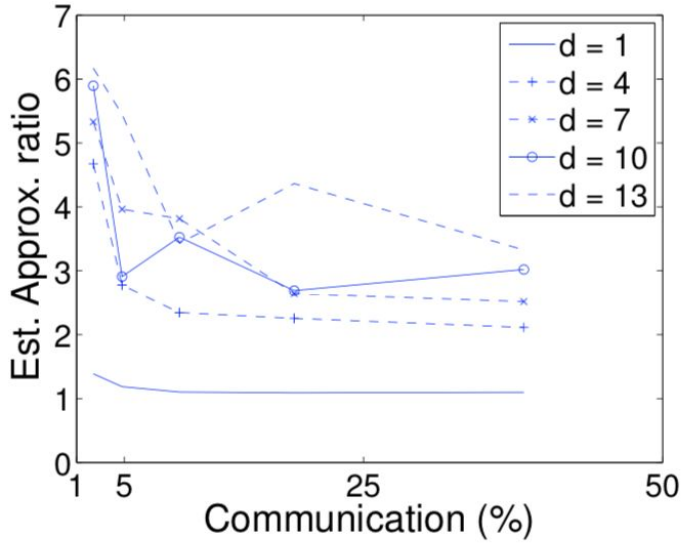


(a) Approx. ratio vs. communication

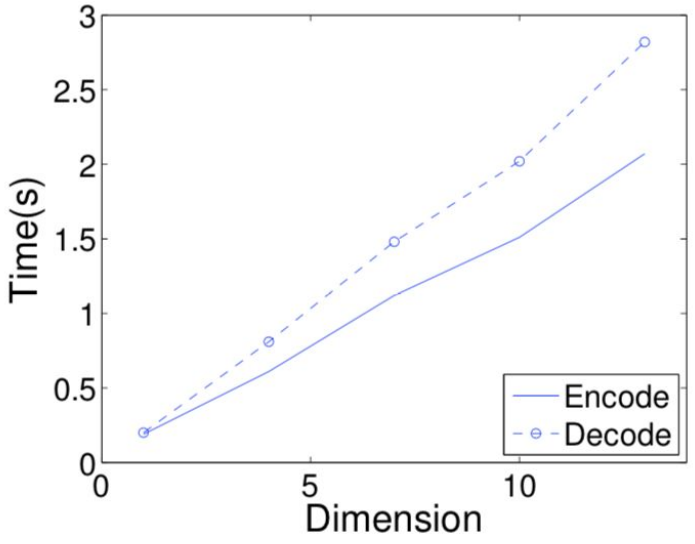


(d) Running time vs. communication

6. Experimental Performance



(a) Approx. ratio vs. dimension



(b) Running time vs. dimension

7. Conclusion

- Protocol can control communication cost and quality (Trade-off)
- Low communication and computation cost
- Single-Round communication
- Tight bound on communication budget

Future Work

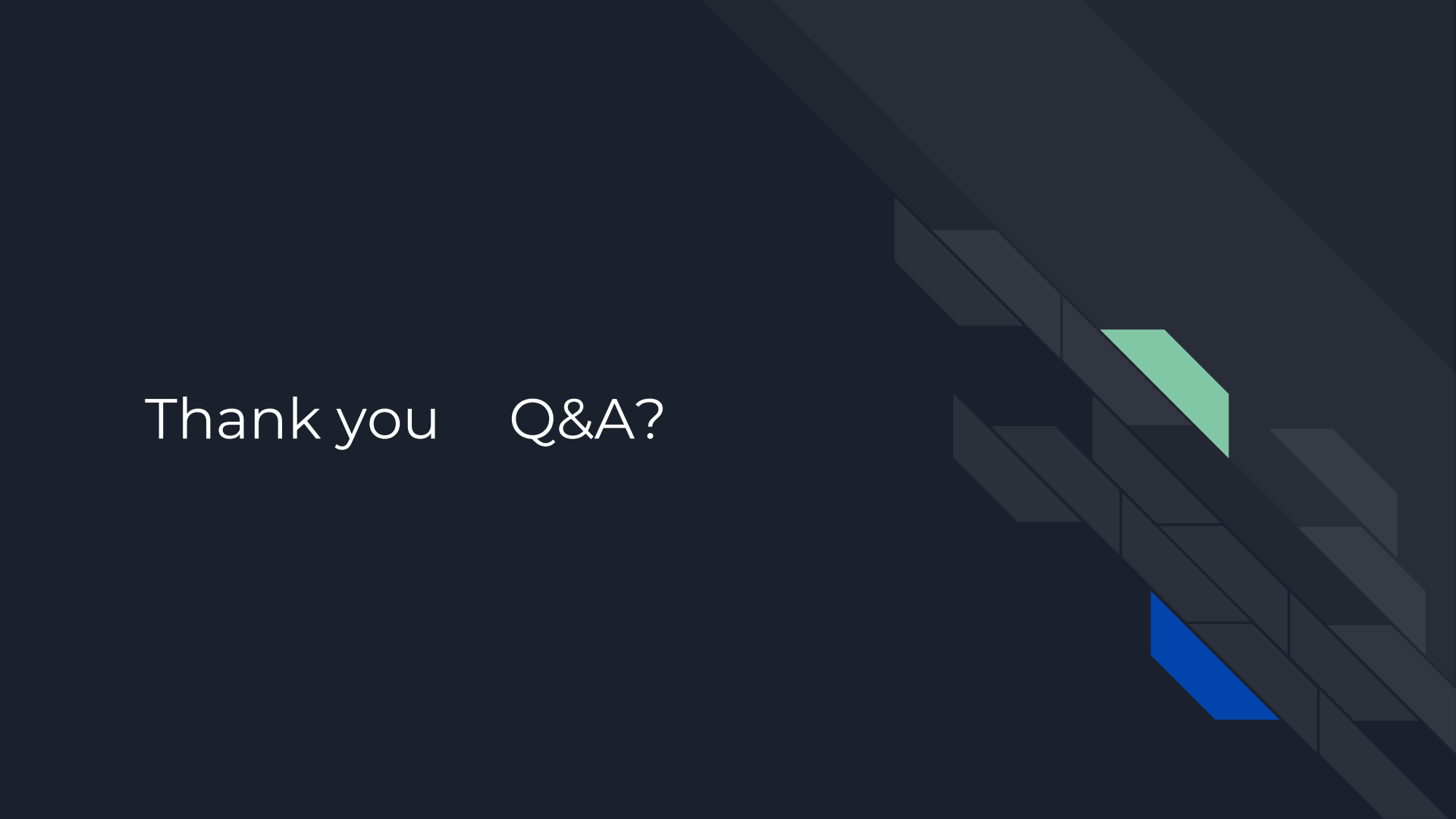
- Possible improvements through multi-round communication
- Resumable reconciliation
- Use of k-d tree to increase accuracy (define size of each grid)
- Estimating set differences before setting budget
- Conversion between budget and IBLT setting
- Comparison to other protocols

References

[1] 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.

[2] Eppstein, David, et al. "What's the difference?: efficient set reconciliation without prior context." ACM SIGCOMM Computer Communication Review. Vol. 41. No. 4. ACM, 2011.

Thank you Q&A?



IBLT

- Table of fixed number of cells with key-value pair (key is a fixed-length integer rep. item)
- A set of hash functions r
- One fingerprint hash function f
- Key = keySum, value = fpSum (fingerprint sum)

Insertion/Deletion:

- Use all r's to designate |r| locations of each key inserted
- Each key is XOR added into keySum
- Use f to compute fingerprint of the key
- Each fingerprint is XOR added into fpSum

Notice the XOR, inserting the same item removes the item

IBLT

Listing entries: (a peeling process)

- Process can start with at least one cell containing one item: f(keysum) = fpsum
- Use the key to retain the item
- Insert the key again to remove it from the table
- Previous step creates more one item cells
- Successfully end with table empty (all zeros)
- Or fail with no more one item cells

IBLT Example

2 hash functions: 2 insert locations

One fingerprint hash function

One difference between sets

IBLT size 3

Alice Key	11100	00011	00100
Key fp	10101	10100	11000
Bob Key	00011	11100	00010
Key fp	10100	10101	01100

keySum	00000	00000	00000
fpSum	00000	00000	00000

IBLT Example

Insert Alice's first key

2 hash functions: 2 locations

Alice Key	11100	00011	00100
Key fp	10101	10100	11000
Bob Key	00011	11100	00010
Key fp	10100	10101	01100

keySum	11100	00000	10101
fpSum	10101	00000	10101

IBLT Example

Insert Alice's second key into IBLT

2nd and 3rd cell

3rd cell is XOR

Alice Key	11100	00011	00100
Key fp	10101	10100	11000
Bob Key	00011	11100	00010
Key fp	10100	10101	01100

keySum	11100	00011	11111
fpSum	10101	10100	00001

IBLT Example

Insert the Last item

Send to Bob

Alice Key	11100	00011	00100
Key fp	10101	10100	11000
Bob Key	00011	11100	00010
Key fp	10100	10101	01100

keySum	11000	00111	11111
fpSum	01101	01100	00001

IBLT Example

Bob Insert (delete) first item into IBLT

Hash function locate item to same cells

Alice Key	11100	00011	00100
Key fp	10101	10100	11000
Bob Key	00011	11100	00010
Key fp	10100	10101	01100

keySum	11000	00100	11100
fpSum	01101	11000	10101

IBLT Example

Insert Bob's second key into IBLT

Order of insertion does not matter

Alice Key	11100	00011	00100
Key fp	10101	10100	11000
Bob Key	00011	11100	00010
Key fp	10100	10101	01100

keySum	00100	00100	00000
fpSum	11000	11000	00000

IBLT Example

Insert the last item from Bob

Alice Key	11100	00011	00100
Key fp	10101	10100	11000
Bob Key	00011	11100	00010
Key fp	10100	10101	01100

keySum	00100	00110	00010
fpSum	11000	10100	01100

IBLT Example

Peeling process to find out Symmetric difference

Evaluate f(keysum) ==fpSum

Alice Key	11100	00011	00100
Key fp	10101	10100	11000
Bob Key	00011	11100	00010
Key fp	10100	10101	01100

keySum	00100	00110	00010
fpSum	11000	10100	01100

IBLT Example

Extract first key from IBLT out

Reinsert again to peel it off

Left with another key

Alice Key	11100	00011	00100
Key fp	10101	10100	11000
Bob Key	00011	11100	00010
Key fp	10100	10101	01100

keySum	00000	00010	00010
fpSum	00000	01100	01100

IBLT Example

We end with an empty IBLT

Retrieved symmetrical difference

$m > (c_k + \varepsilon)t$ m cells, t differences, ck = m/t

$P(fail) = O(t^{-k}+2)$ k num of hash functions

Alice Key	11100	00011	00100
Key fp	10101	10100	11000
Bob Key	00011	11100	00010
Key fp	10100	10101	01100

keySum	00000	00000	00000
fpSum	00000	00000	00000