Graphene: Efficient Interactive Set Reconciliation Applied to Blockchain Propagation

Pinar Ozisik
Brian Levine
George Bissias
Gavin Andresen
Darren Tapp
Sunny Katkuri

UMass Amherst
Women for UMass Amherst
ASU
• P2p distributed systems that are unstructured

• End-points need to talk to each other
• Transactions (txns) are transfers of money
Background: Txns

- Transactions (txns) are transfers of money
- Unvalidated txns are broadcast
Background: Blocks

- Transactions (txns) are transfers of money
- Unvalidated txns are broadcast
Transactions (txns) are transfers of money

Unvalidated txns are broadcast
Background: Blocks

- Transactions (txns) are transfers of money
- Unvalidated txns are broadcast
- Blocks are comprised of txns
Background: Blocks

- Blocks are comprised of txns
Background: Blocks

• Blocks are comprised of txns
• Blocks are broadcast
Background: Mempool

- Blocks are comprised of txns
- Each peer in the network has a pool of unvalidated txns, called the mempool
• Blocks are comprised of txns

• Each peer in the network has a pool of unvalidated txns, called the mempool

Background: Mempool
Background: Mempool

- Blocks are comprised of txns
- Each peer in the network has a pool of unvalidated txns, called the mempool
- Peers clear out txns from their mempool
Setup
Setup

• **Goal**: Send as little data as possible over the wire
Motivation

• Many distributed systems require synchronization of records among processes

• Blockchains are just the latest example
  • Replicas in distributed databases
  • Distributed sensors
  • Security certifications

• Must solve **set reconciliation**
Hard Forks

• Maximum block size is a network-wide parameter

• Block size was the catalyst for the initial fork of Bitcoin Cash from Bitcoin
Hard Forks

- Maximum block size is a network-wide parameter
- Block size was the catalyst for the initial fork of Bitcoin Cash from Bitcoin
Problem Definition

• Given a block of txns from Alice, and a set of txns at Bob, determine:

• The subset of Bob’s txns that are in the block
Problem Definition

• Given a block of txns from Alice, and a set of txns at Bob, determine:

  • The subset of Bob’s txns that are in the block
  • The subset of txns that Bob is missing
Contributions

• In blockchains, often Bob has a superset of the block
  • We solve this problem in one roundtrip
  • But what if he doesn’t? We also solve that case in a followup round

• We use two probabilistic data structures:
  • Bloom filters (BFs)
  • Invertible Bloom Lookup Tables (IBLTs)

• We need guaranteed performance
Contributions

• **A new protocol** that solves which elements in a set $M$ stored by a receiver are members of a subset $N \subseteq M$

• **Extension of our protocol** where some of the elements of $N$ are missing

• **Efficient search algorithm for parameterizing an IBLT**

• Evaluation using **open-source deployment** in the real-world, mathematical analysis, and **simulation**
Problem Setup

• We want to avoid sending a lot of data and relay information quickly

• Assumption: the block is a subset of the mempool
  • $n$ txns in the block
  • $m$ txns in the mempool
Compact Blocks

- Don’t need to send full txns
- Can send hash of each txn
- This has been deployed
Bloom Filters

• Bloom filters represent a set of items

• They are useful for set membership

• Special feature:
  • **Membership test returns negative:** item is a True Negative and not part of the set
  • **Membership test returns positive:** item is a True Positive or a False Positive
Bloom Filters

- The False Positive Rate (FPR) is tunable
- More bits will lower the FPR

Number of FPs we will observe approximately follow a binomial distribution with two parameters:
- \( n \): number of items to test for membership
- \( p \): probability of failure
Bloom Filters

- Based on mempool, not block
- If FPR = 1/m, then we expect 1 transaction from mempool to falsely appear to be in the block
- Block reconstruction will fail every block!

\[
\begin{align*}
\text{Alice:} & \quad \text{has block} \\
\text{Bob:} & \quad \text{wants block} \\
\text{mempool:} & \quad \text{getdata, } m
\end{align*}
\]

\[
S = \text{Bloom(txns)}
\]

FPR = 1/m

block = mempool found in S
Invertible Bloom Lookup Tables

- IBLTs are a generalization of Bloom Filters
  - Instead of a bit, cells include a count and actual content

- IBLTs support **subtraction**
  - IBLTs must be **the same size** for subtraction
  - Subtraction recovers **symmetric difference**
  - If subtraction recovers the entire symmetric difference, then we say that the **subtraction decoded**

\[
\begin{align*}
A, B, C, \\
D, E, F
\end{align*}
\]

\[
A, B, C, \\
\Delta \quad \X, E, F
\]

\[
\begin{align*}
\Delta & = + D \\
& - X
\end{align*}
\]

Goodrich and Mitzenmacher; Eppstein et al.
Invertible Bloom Lookup Tables

- IBLTs are a generalization of Bloom Filters
  - Instead of a bit, cells include a count and actual content

\[ \Delta \]

\[
\begin{align*}
A, B, C, D, E, F \\
A, B, C, X, E, F \\
\end{align*}
\]

- Special IBLT feature:
  - The size of IBLTs does not depend on the original sets
  - It depends on the size of the expected difference between the two sets

Goodrich and Mitzenmacher; Eppstein et al.
• The size of the IBLT will depend on the symmetric difference between the block and the receiver’s mempool

• Based on mempool as well!
Graphene

• It’s costly to send Bloom Filters when mempool is large
• It’s costly to send IBLTs when mempool is large

• Solution:
  • **Use a Bloom Filter** to reduce the symmetric difference between block and mempool
  • **Use an IBLT** to recover from small errors in the Bloom Filter
Why two data structures?

- There is something to optimize!
- The figure shows that we don’t need a low FPR for Bloom filter
- The IBLT can help us recover from the mistakes made by the Boom filter
Graphene: Protocol 1

- The block is a subset of the mempool
Graphene: Protocol 1

- The block is a subset of the mempool

Sender

Receiver

block
Graphene: Protocol 1

- The block is a subset of the mempool
Graphene: Protocol 1

- The block is a subset of the mempool

Sender

Receiver

block

IBLT

Bloom Filter S
Graphene: Protocol 1

- The block is a subset of the mempool

Sender

Receiver

block

IBLT

Bloom Filter $S$
Graphene: Protocol 1

- The block is a subset of the mempool
Graphene: Protocol 1

- The block is a subset of the mempool

Sender

Receiver

block

IBLT

true negatives
Graphene: Protocol 1

- The block is a subset of the mempool

Sender

Receiver

block

IBLT \( I \) \quad \Delta \quad IBLT \( I' \)

true negatives
Graphene: Protocol 1

- The block is a subset of the mempool

Sender

Receiver

block

$\Delta$

true positives

true negatives

Sender

Receiver

$IBLT_I$ $\Delta$ $IBLT_{I'}$
Graphene: Protocol 1

- The block is a subset of the mempool

Sender

Receiver

block

IBLT \( I \)

IBLT \( I' \)

in block

NOT in block
Graphene: Protocol 1

- The block is *not* a subset of the mempool
Graphene: Protocol 1

- The block is *not* a subset of the mempool

Sender

block

Receiver

missing txns!

in block

NOT in block
Graphene: Protocol 1

• The block is *not* a subset of the mempool

Sender

Receiver

missing txns!

Block

Bloom Filter $S$

IBLT $I$

in block

NOT in block

Sender

Receiver
Graphene: Protocol 1

- The block is *not* a subset of the mempool

Sender

Receiver

true negatives

missing txns!
Graphene: Protocol 1

- The block is *not* a subset of the mempool
Graphene: Protocol 2

• The block is *not* a subset of the mempool

Sender

Receiver

missing txns!

true negatives
Graphene: Protocol 2

- The block is *not* a subset of the mempool

Sender

Block

Receiver

Bloom Filter $R$

true negatives

missing txns!
Graphene: Protocol 2

- The block is *not* a subset of the mempool

Sender

Receiver

Bloom Filter $R$

missing txns!

true negatives
Graphene: Protocol 2

- The block is *not* a subset of the mempool

Sender

Receiver

block
Bloom Filter $R$

true negatives

missing txns!
Graphene: Protocol 2

- The block is *not* a subset of the mempool

**Block**

Sender

Bloom Filter $R$

**Receiver**

missing txns!

ture negatives
Graphene: Protocol 2

- The block is *not* a subset of the mempool

**Sender**

**Receiver**

**Block**

Bloom Filter $R$

**IBLT**

$J$

true negatives

missing txns!
Graphene: Protocol 2

- The block is *not* a subset of the mempool

Sender

Receiver

**Block**

Bloom Filter $R$

**IBLT**

$J$

**IBLT**

$J'$

$\Delta$

true negatives

missing txns!
Graphene: Protocol 2

• The block is not a subset of the mempool

Sender

Receiver

block
Bloom Filter \( R \)

IBLT \( J \) \( \Delta \) IBLT \( J' \)

true negatives

missing txns!
• The block is *not* a subset of the mempool
Graphene: Protocol 2

- The block is *not* a subset of the mempool

Sender

Receiver

**block**

Bloom Filter $R$

IBLT $J$

$\Delta$

IBLT $J'$

in block

NOT in block
• **Two challenges** with IBLTs:
  • Estimating the size of the symmetric difference so subtraction works
  • Given that we *know* the symmetric difference, what is the *smallest* size that can recover the items?

• No such formula for IBLTs, only asymptotic results
IBLTs

• Binary search over hypergraphs
• Hypergraphs are an equivalent representation of IBLTs
• Larger but computation is faster
• Create a universal lookup table
Open-Source Deployment

• Deployed on Bitcoin Cash network via the Bitcoin Unlimited client
  • 1,431 nodes

• Fraction of the size of previous work

• Deploying a protocol requires real engagement with the community

• Adversarial thinking is critical

• Mempools are in-sync less often than expected
Evaluation

- Three block sizes in terms of number of txns
- The receiver’s mempool contains:
  - All transactions in the block
  - Additional txns as a multiple of the block size
- Improvement with block size
Evaluation

- Three block sizes in terms of number of txns
- The receiver’s mempool contains
  - All transactions in the block
  - Additional txns as a multiple of the block size
- Improvement with block size
Summary

• **Key points**
  • Presented Protocol 1 & 2
  • Showed how to parameterize IBLTs
  • Presented deployment and simulation results

• **Public implementation**
  • Graphene in BCH client: https://github.com/BitcoinUnlimited/BitcoinUnlimited
  • Stand alone IBLT implementation: https://github.com/umass-forensics/IBLT-optimization
  • Graphene simulations: https://github.com/pinaro/Graphene-simulations

• **Contacts:**
  • Pinar Ozisik: pinar@cs.umass.edu
  • Brian Levine: bnl@umass.edu
  • George Bissias: gbiss@cs.umass.edu

Thank you!