Graphene: Efficient Interactive Set Reconciliation Applied to Blockchain Propagation

Pinar Ozisik Brian Levine George Bissias

Gavin Andresen Darren Tapp Sunny Katkuri

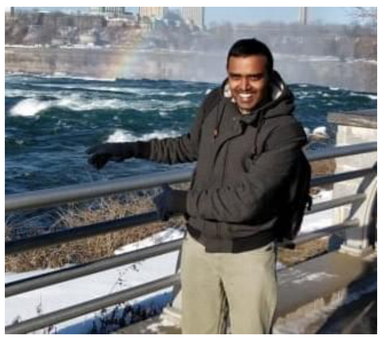
UMassAmherst

Women UMASS for AMHERST

ASU

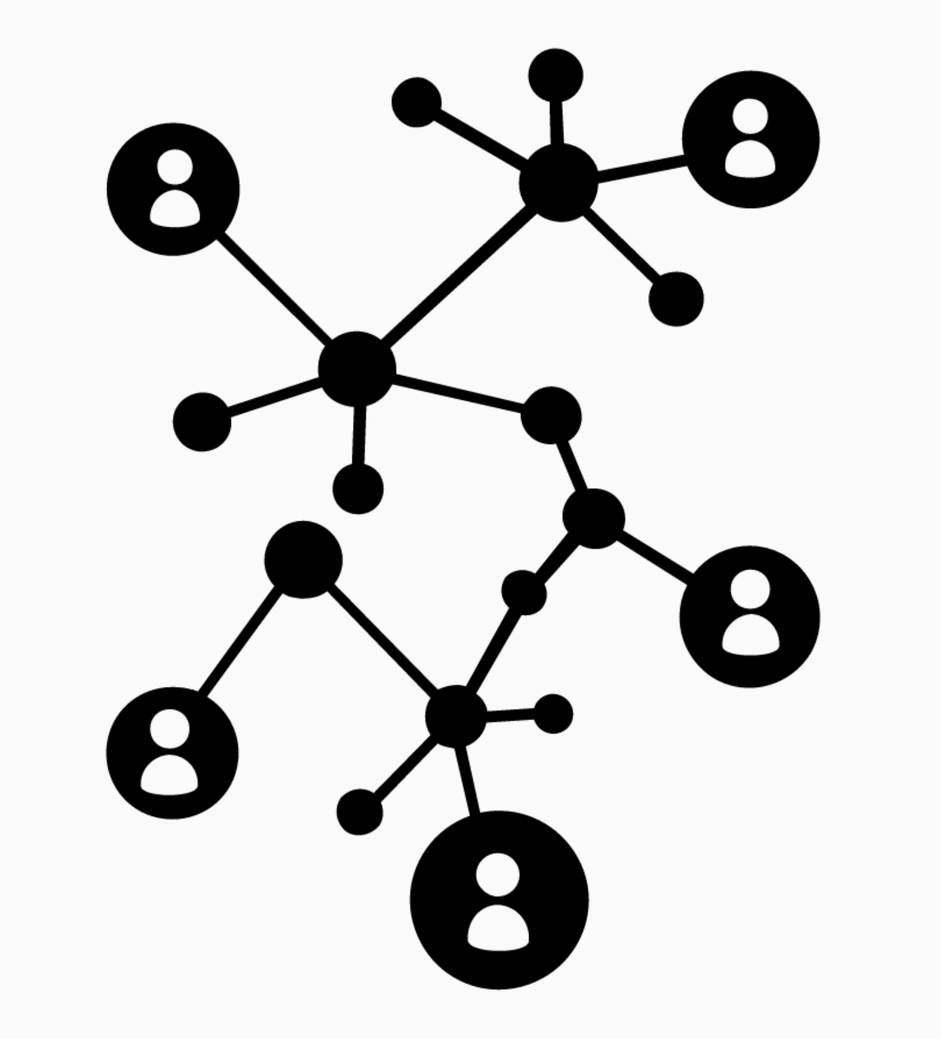




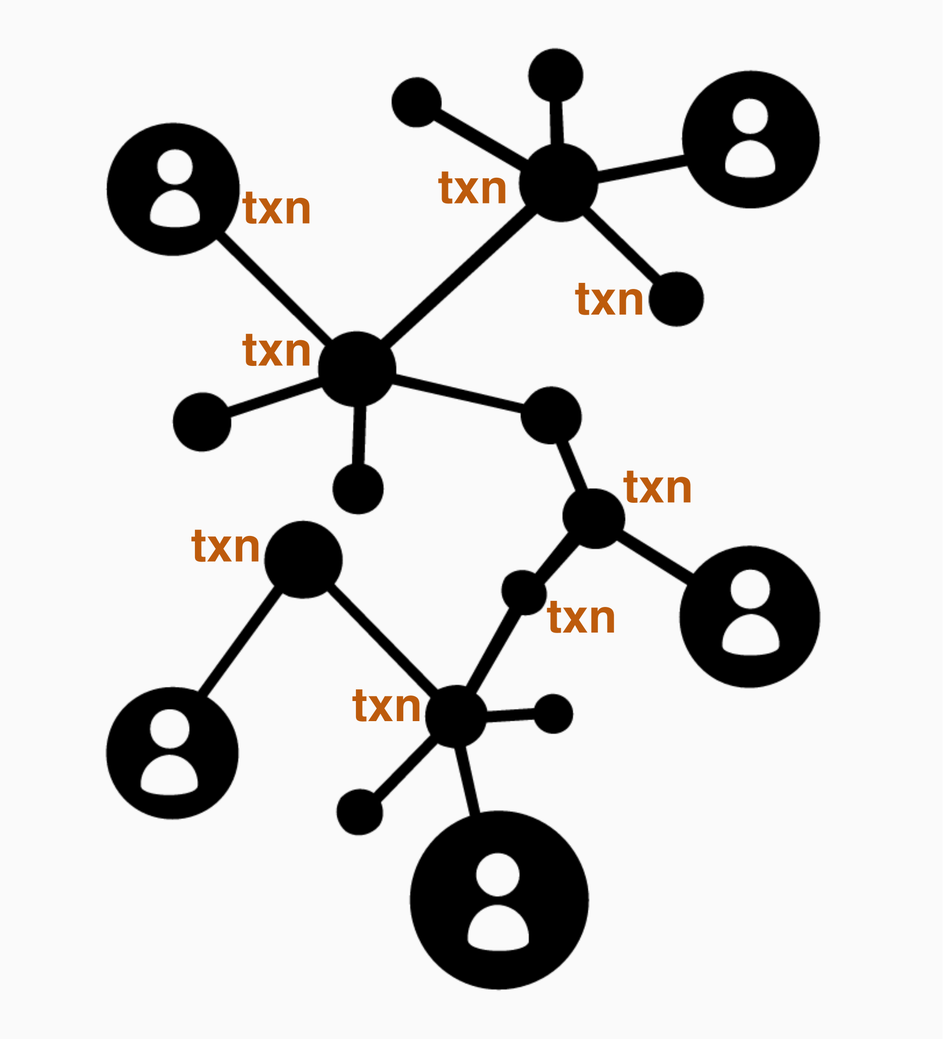
Background

- P2p distributed systems that are unstructured
- End-points need to talk to each other



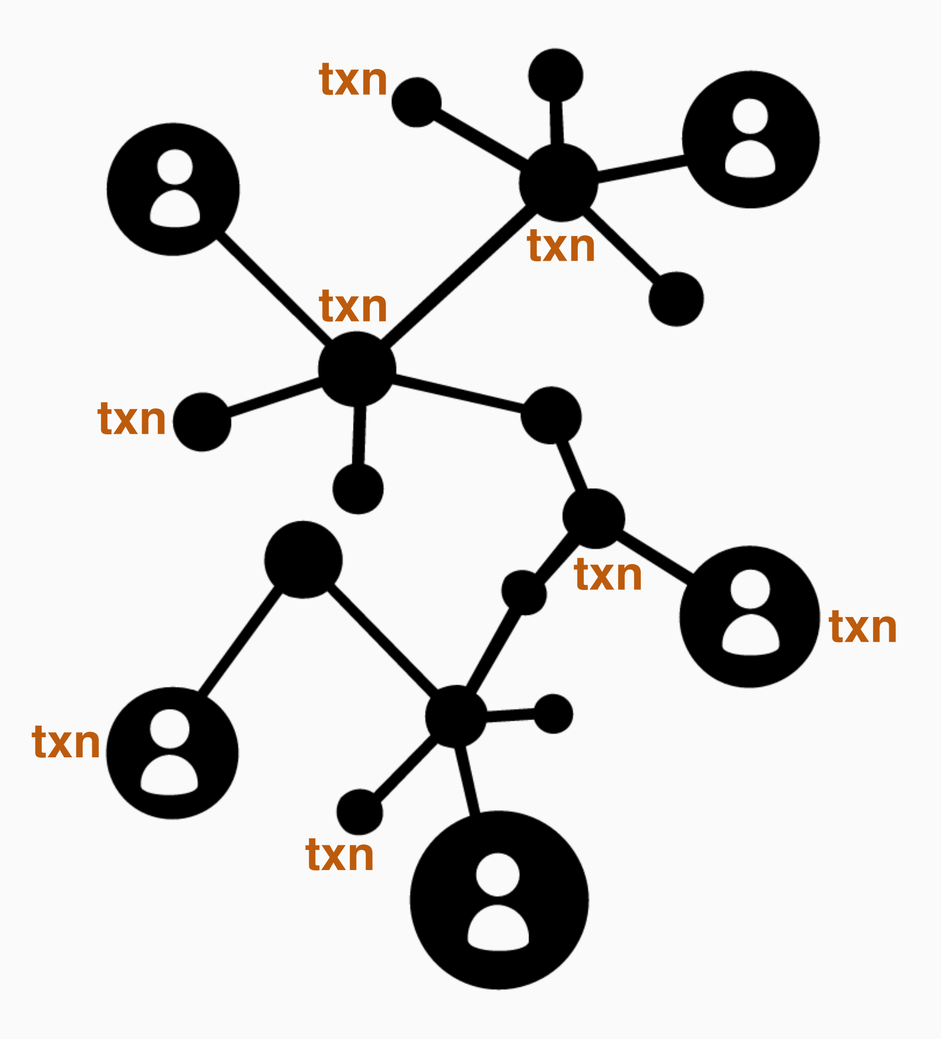
Background: Txns

- Transactions (txns) are transfers of money

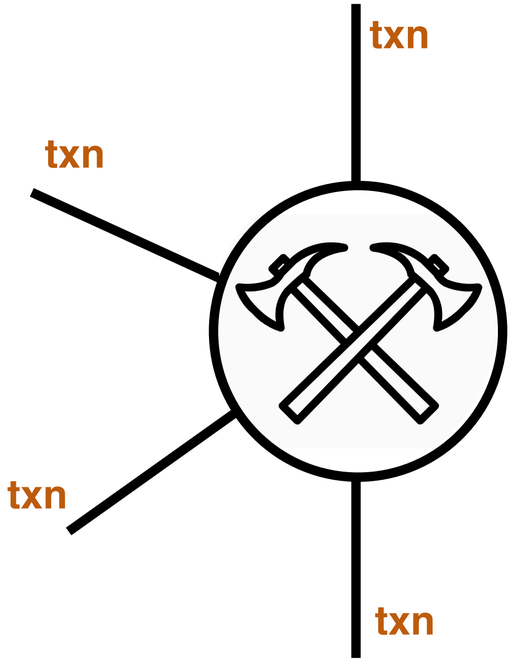


Background: Txns

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

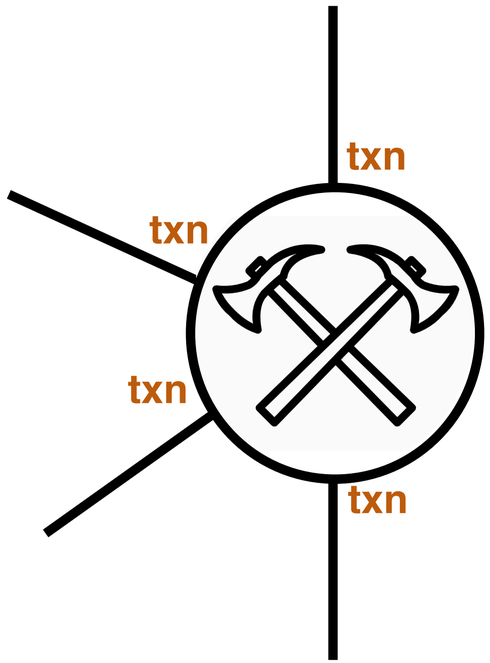


Background: Blocks



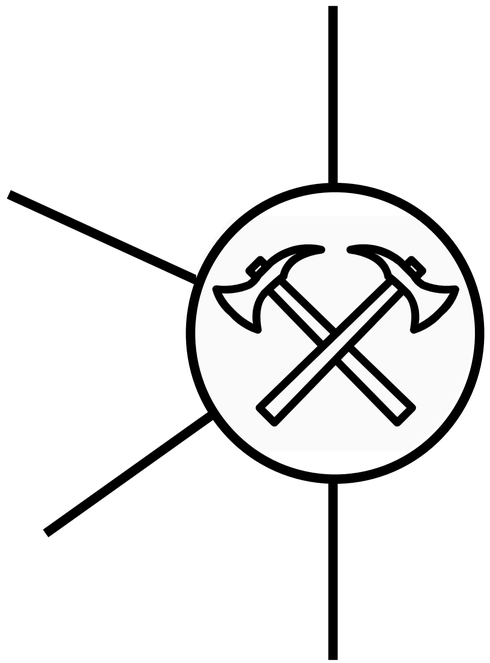
- Transactions (txns) are transfers of money
- Unvalidated txns are broadcast

Background: Blocks



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

Background: Blocks

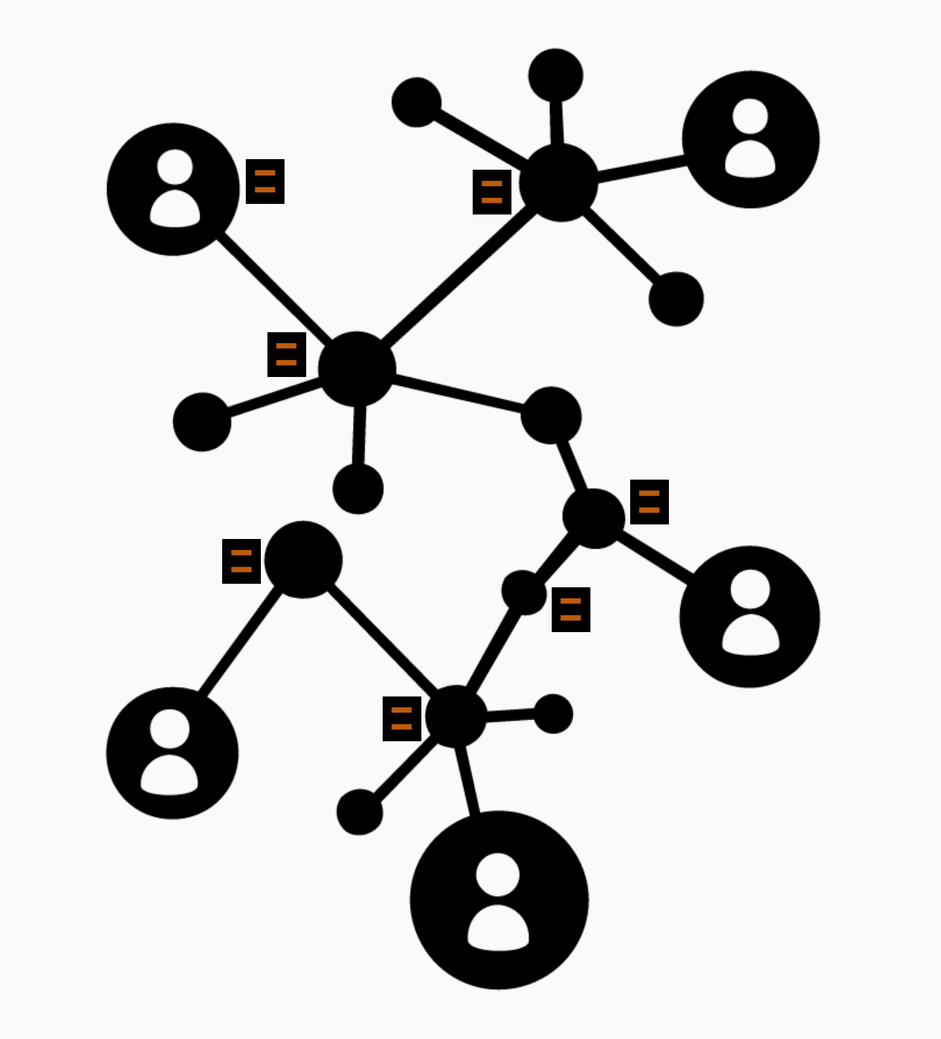


Block

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

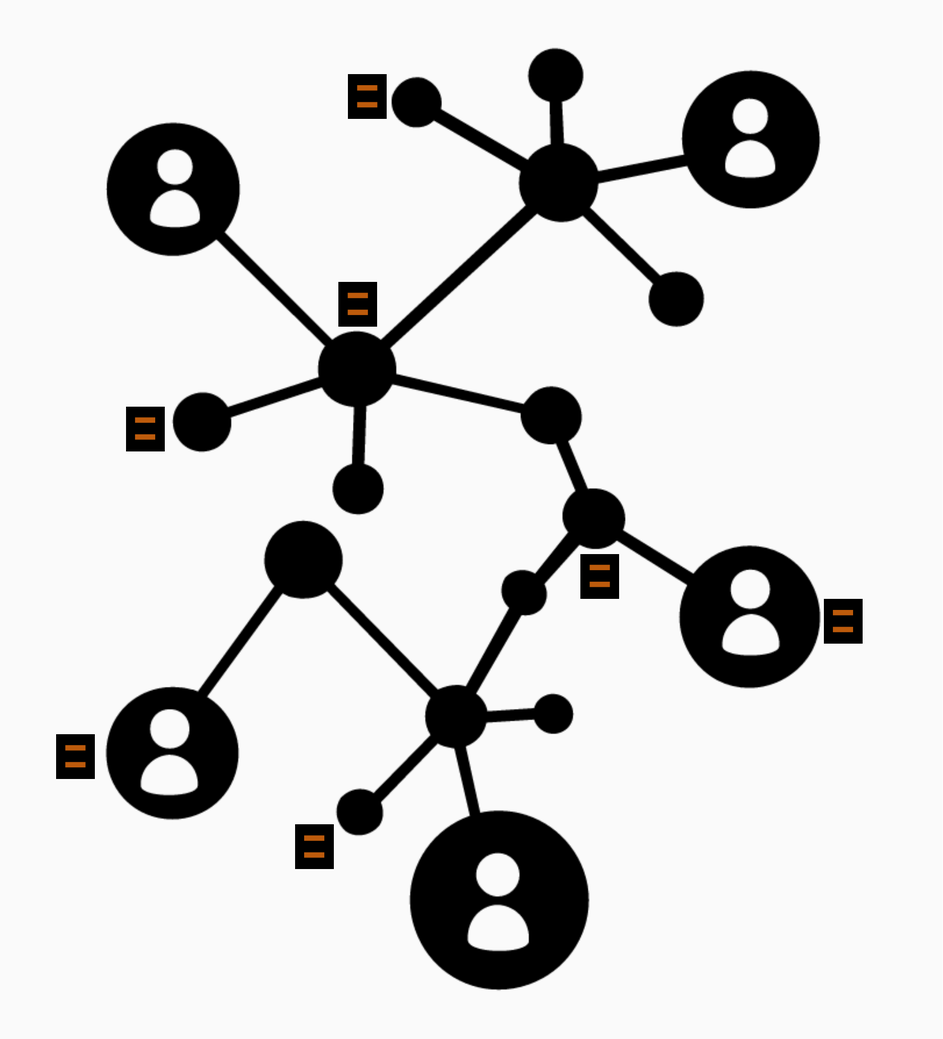
Background: Blocks

- Blocks are comprised of txns

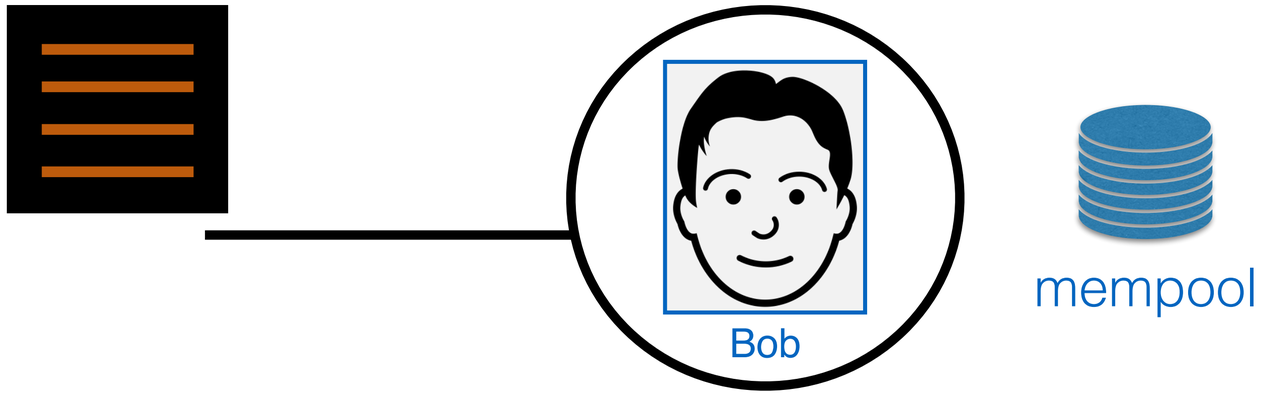


Background: Blocks

- Blocks are comprised of txns
- Blocks are broadcast

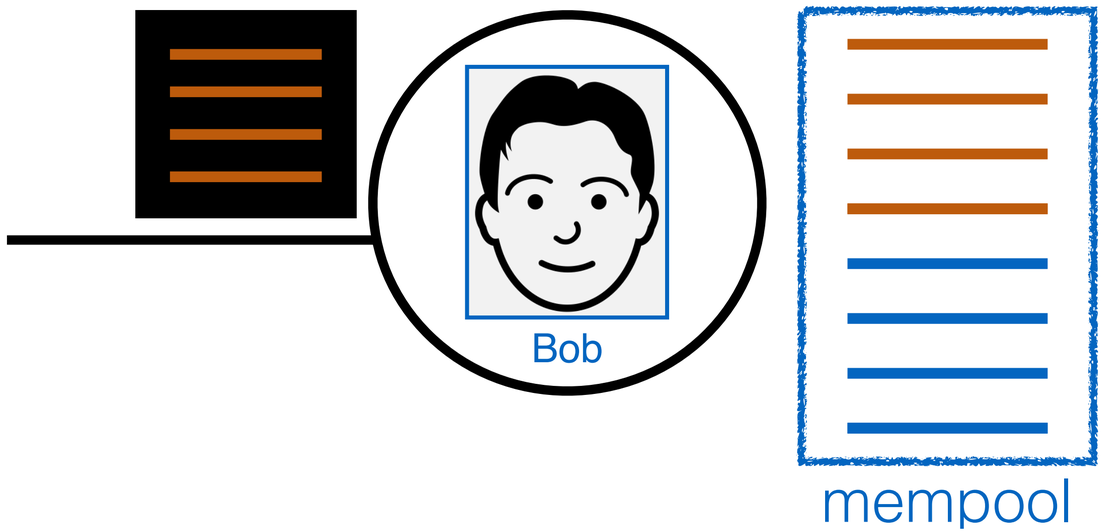


Background: Mempool



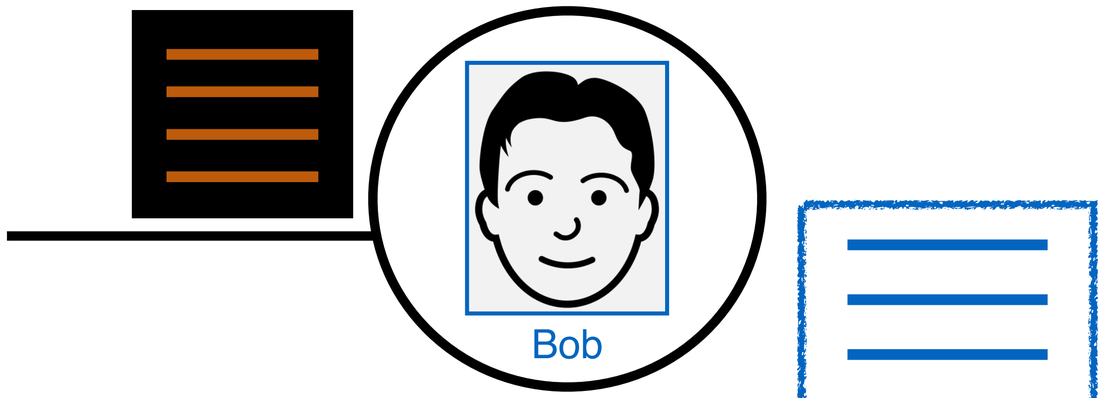
- Blocks are comprised of txns
- Each peer in the network has a pool of unvalidated txns, called the **mempool**

Background: Mempool



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

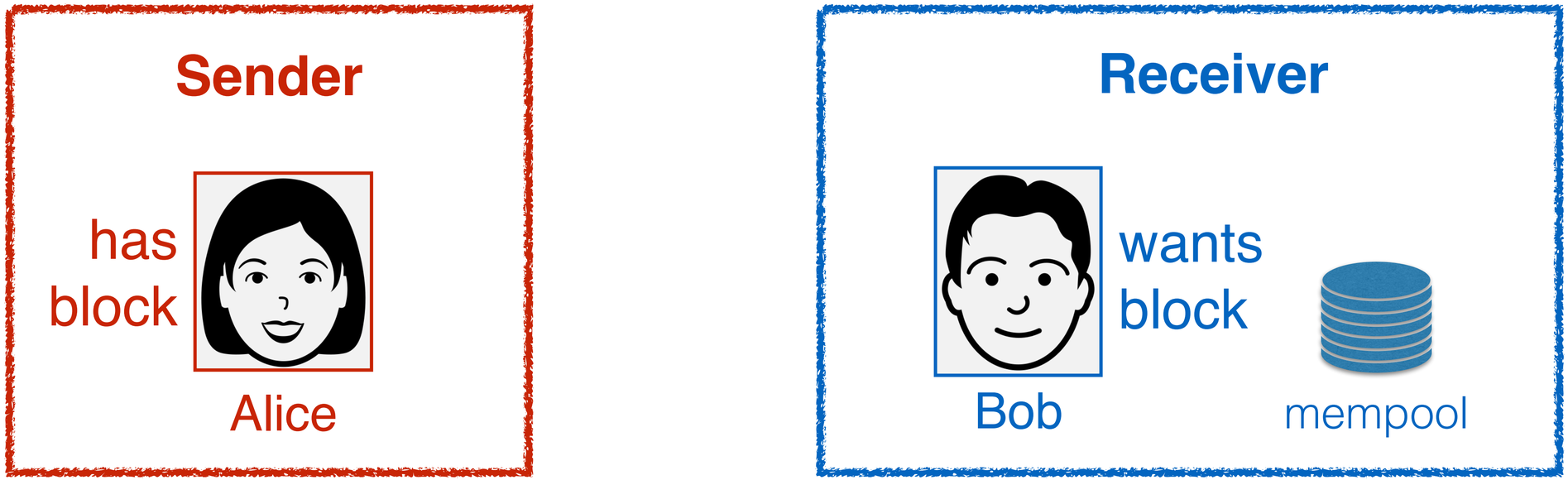
Background: Mempool



- 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

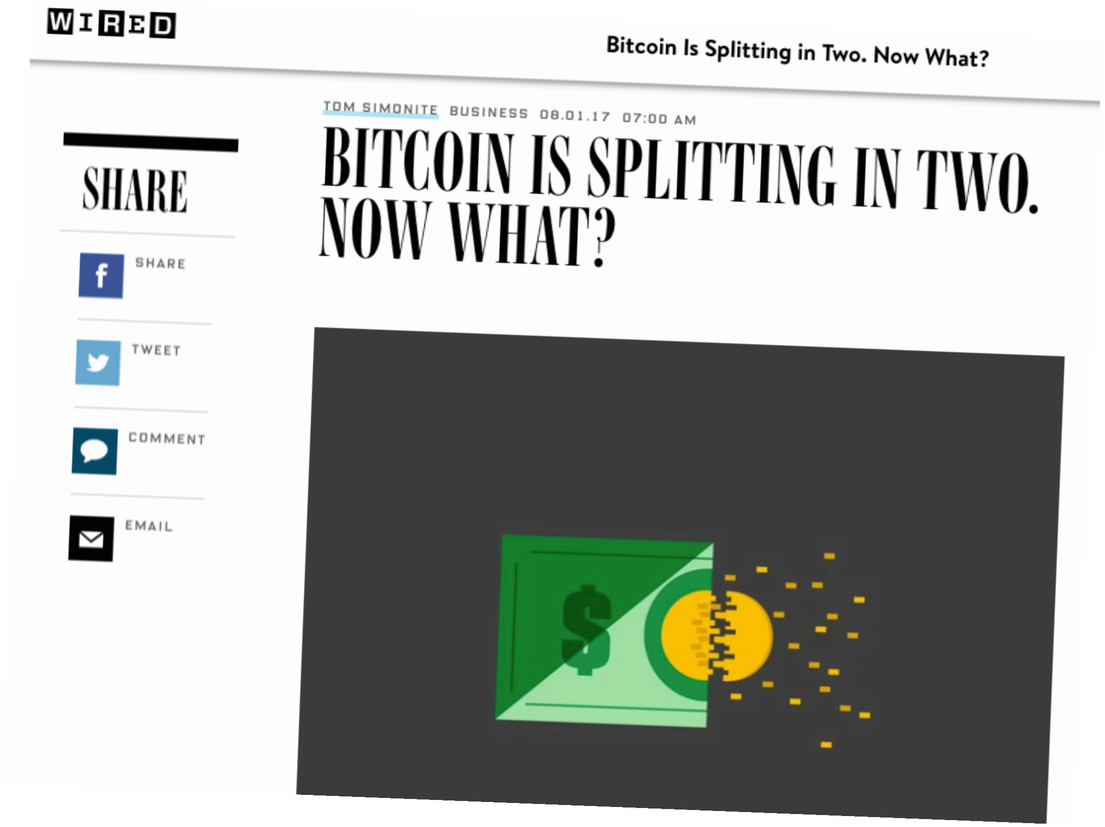
- Many distributed systems require synchronization of records among processes

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

- Must solve **set reconciliation**

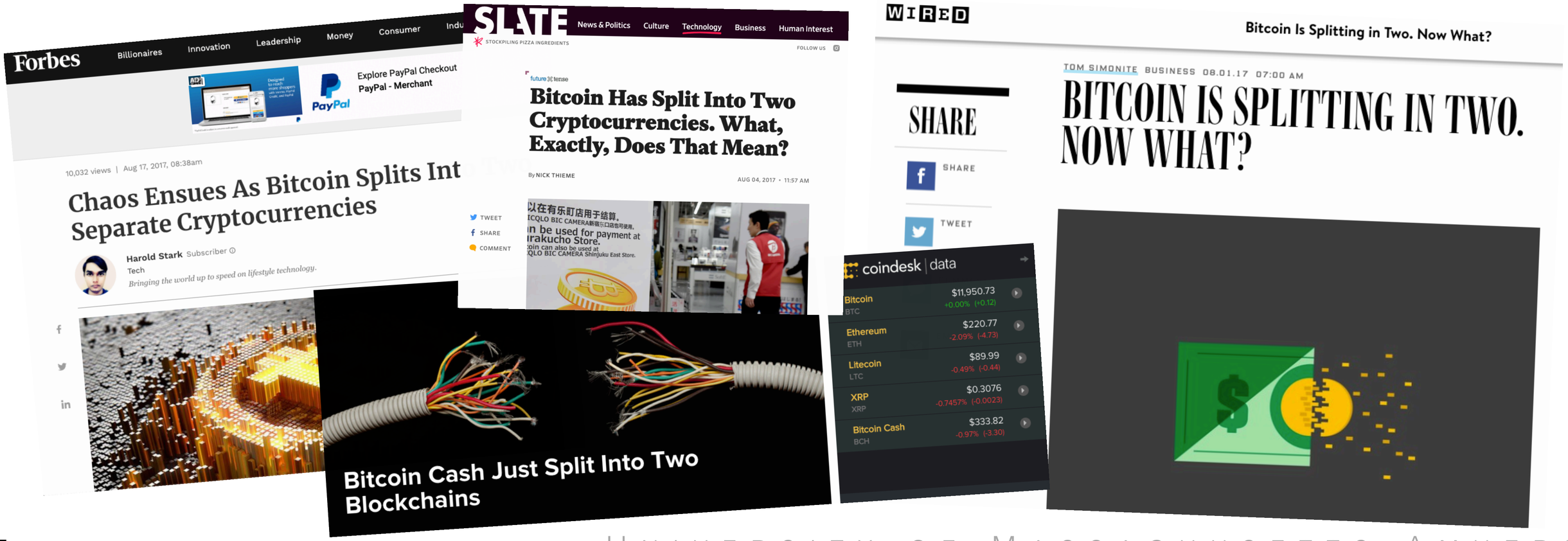
Hard Forks

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



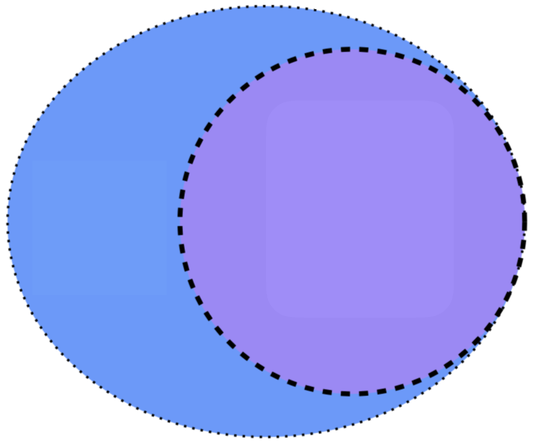
Hard Forks

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



Problem Definition

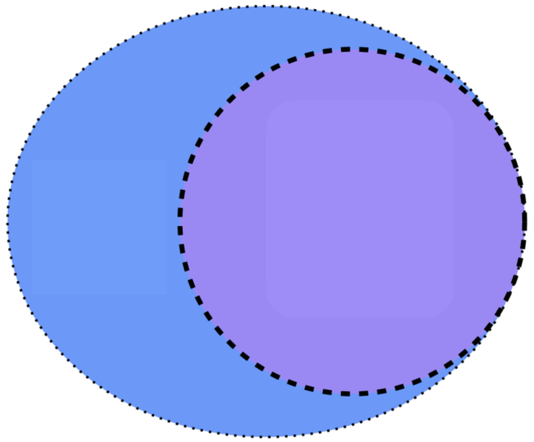
- Given a block of txns from Alice, and a set of txns at Bob, determine:



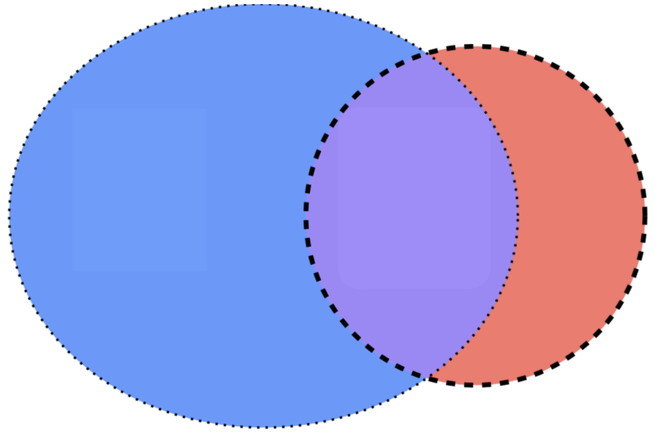
- The subset of Bob's txns that are in the block

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**

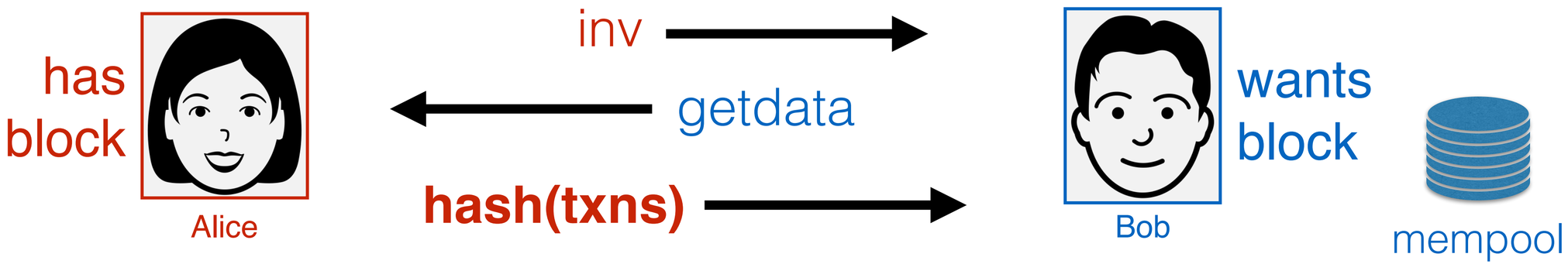
Problem Setup

- 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

B. Bloom

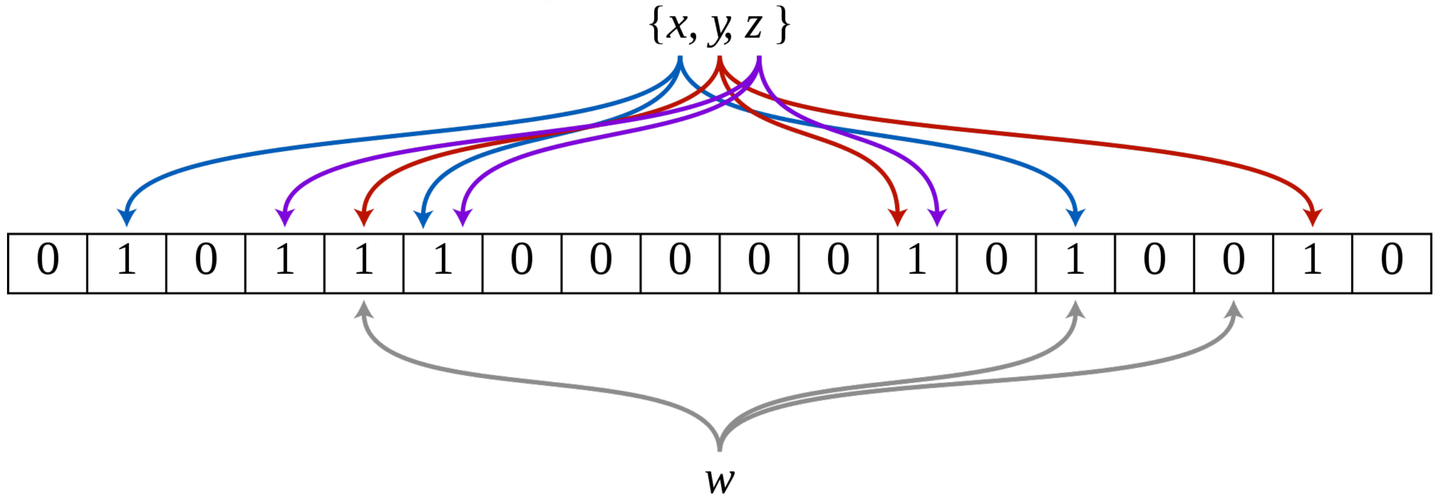
- Bloom filters represent a set of items
- They are useful for set membership
- Special feature:
  - **Membership test returns negative:** item is a True Negative and not part of the set
  - **Membership test returns positive:** item is a True Positive or a False Positive

Bloom Filters

B. Bloom

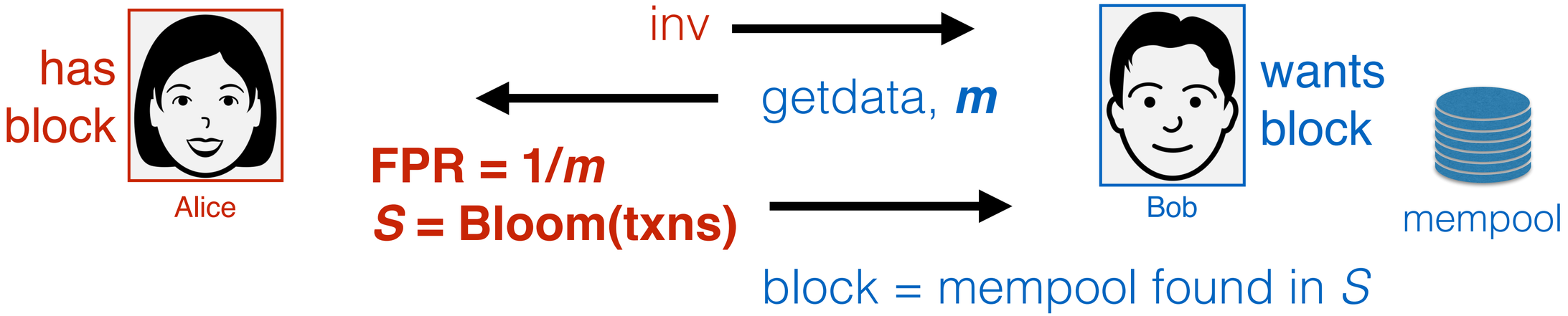
- The False Positive Rate (FPR) is tunable
  - More bits will lower the FPR

- 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

B. Bloom

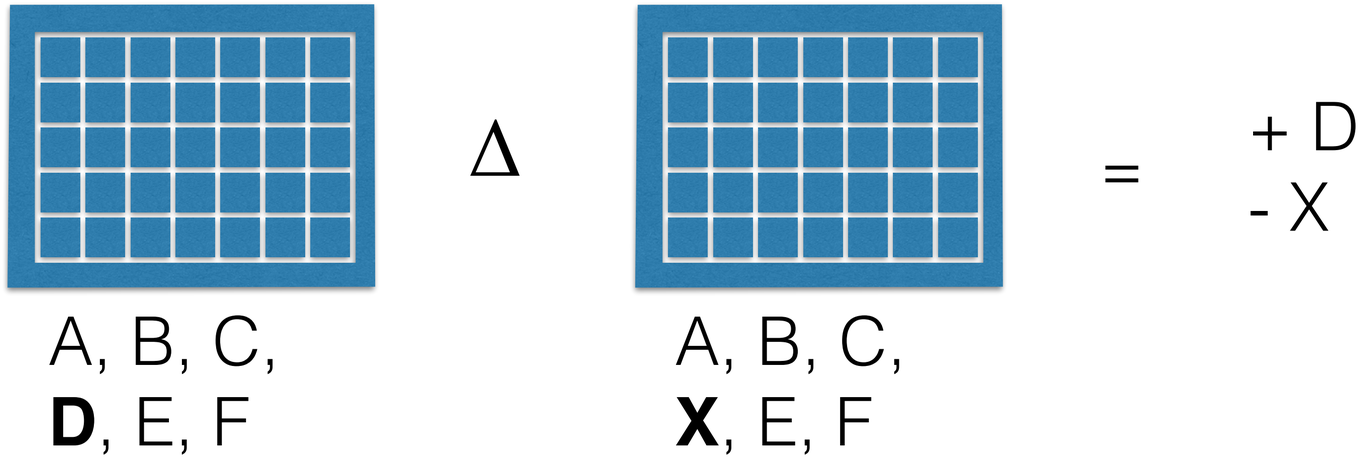


- Based on mempool, not block
- If **FPR = 1/*m***, then we expect 1 transaction from mempool to falsely appear to be in the block
- Block reconstruction will fail every block!

Invertible Bloom Lookup Tables

Goodrich and Mitzenmacher; Eppstein et al.

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

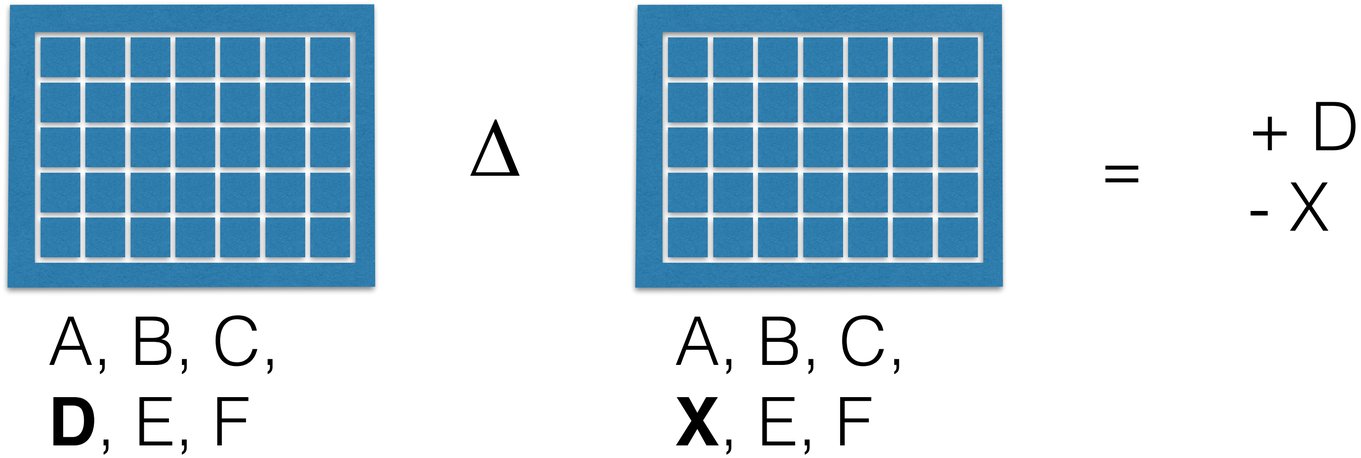


- IBLTs support **subtraction**
  - IBLTs must be **the same size** for subtraction
  - Subtraction recovers **symmetric difference**
- If subtraction recovers the entire symmetric difference, then we say that the **subtraction decoded**

Invertible Bloom Lookup Tables

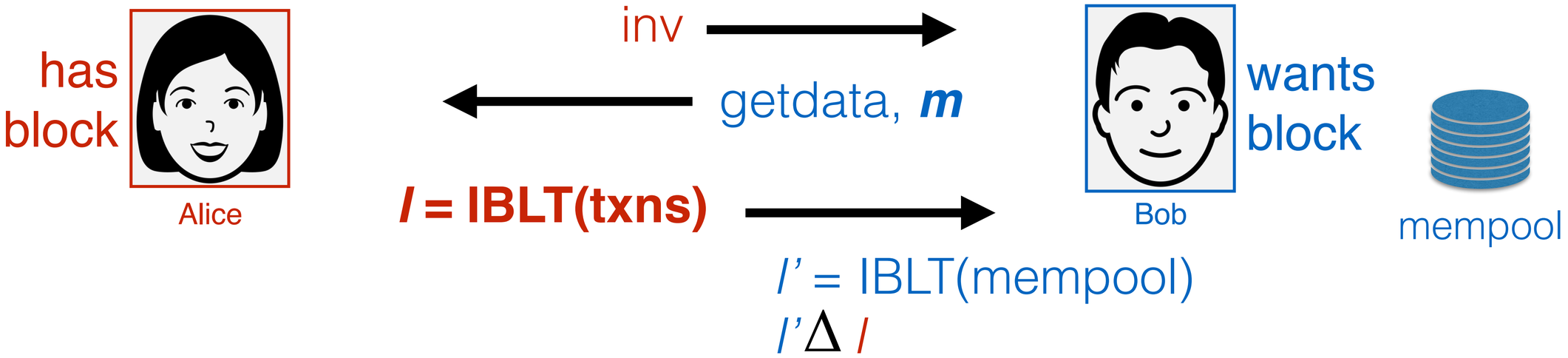
Goodrich and Mitzenmacher; Eppstein et al.

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



- 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

IBLTs



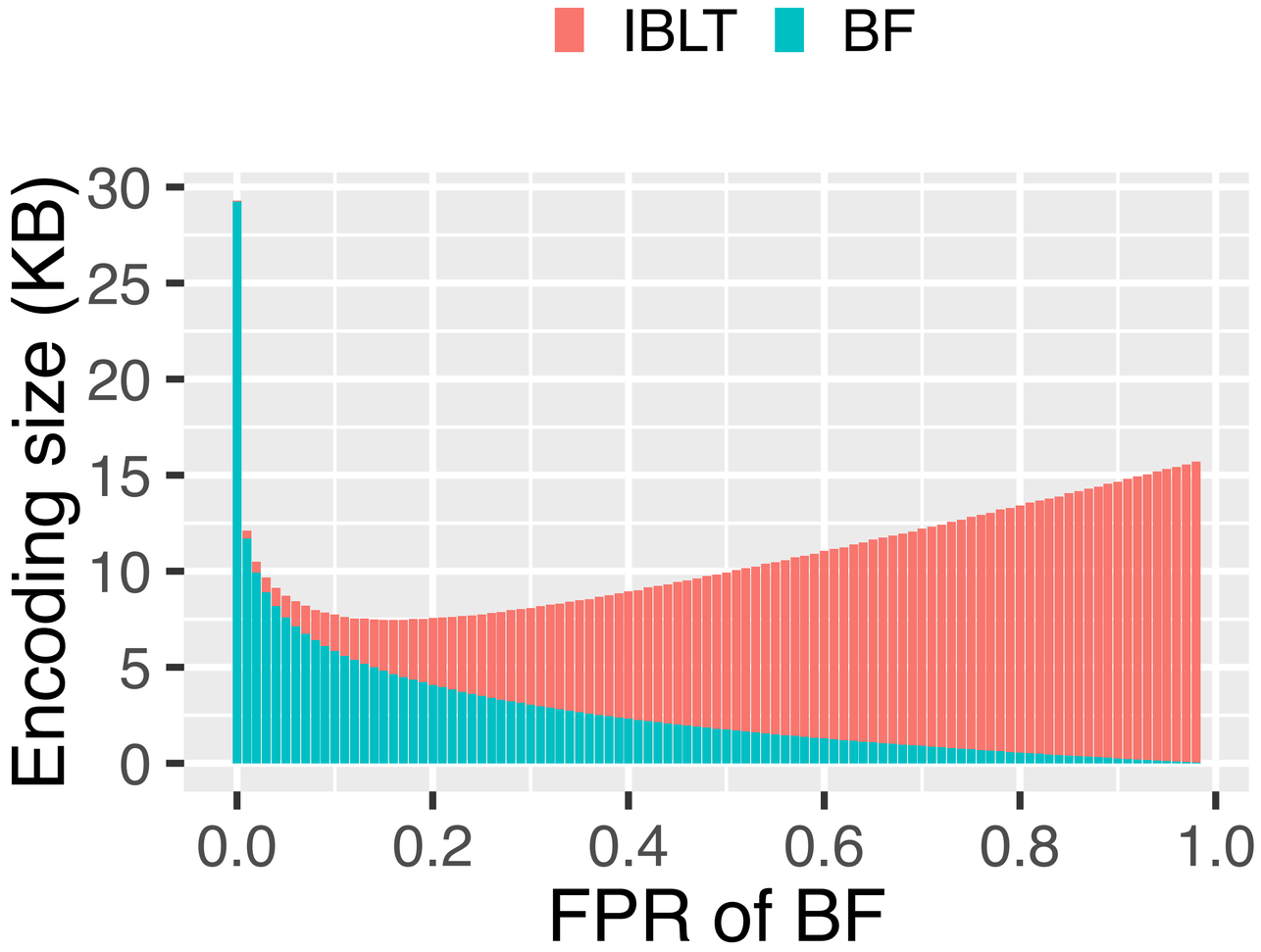
- The size of the IBLT will depend on the symmetric difference between the block and the receiver's mempool
- Based on mempool as well!

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

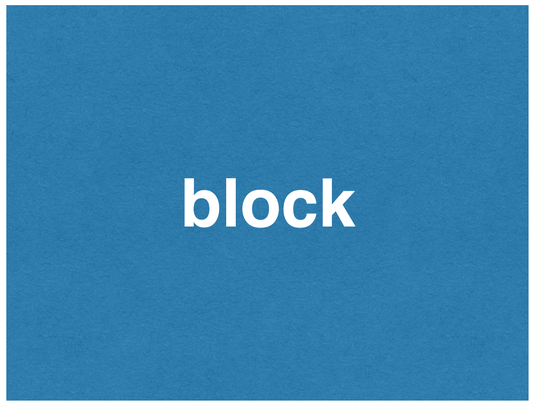


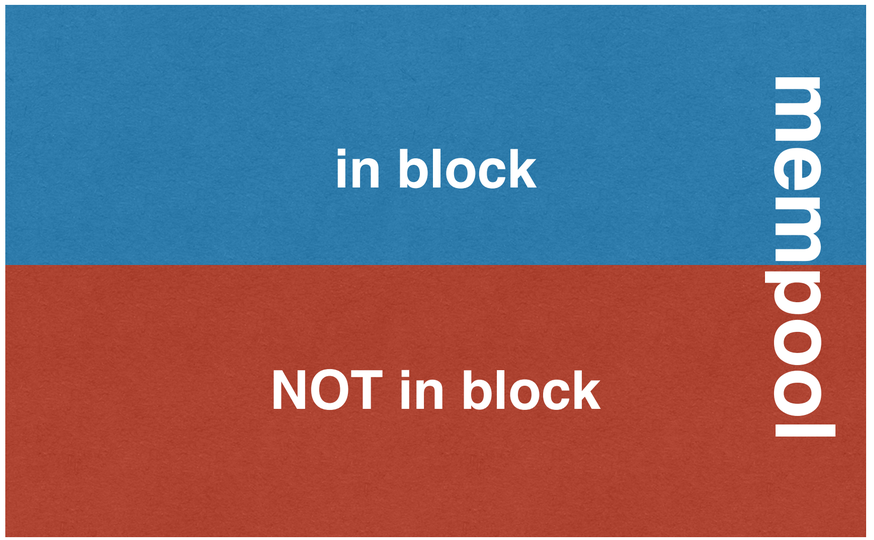
Graphene: Protocol 1

- The block is a subset of the mempool

Sender 

Receiver 



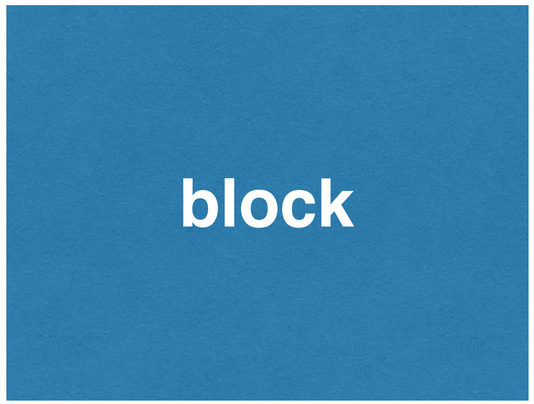


Graphene: Protocol 1

- The block is a subset of the mempool

Sender 

Receiver 

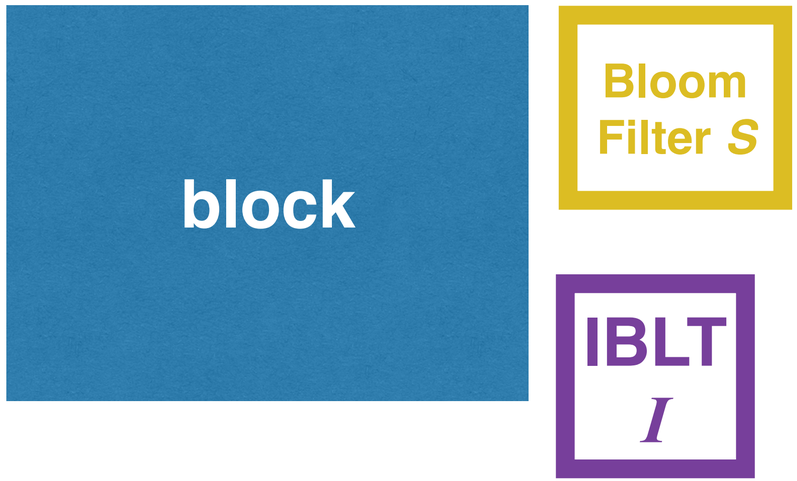


Graphene: Protocol 1

- The block is a subset of the mempool

Sender 

Receiver 

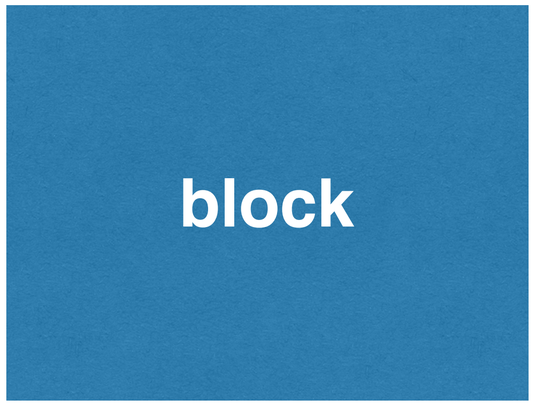


Graphene: Protocol 1

- The block is a subset of the mempool

Sender 

Receiver 

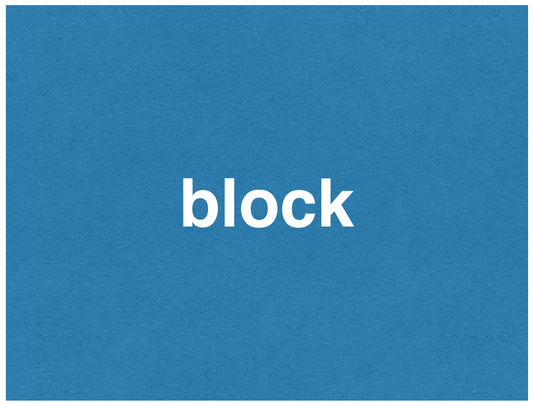


Graphene: Protocol 1

- The block is a subset of the mempool

Sender 

Receiver 

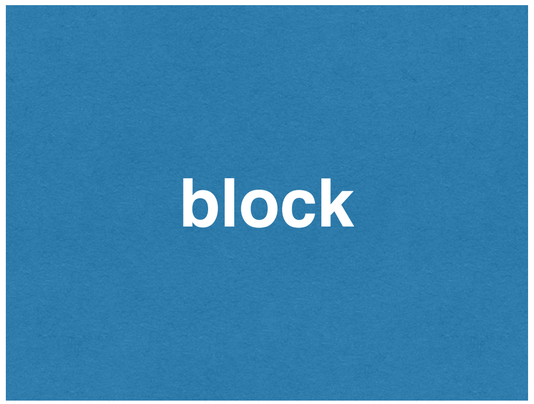


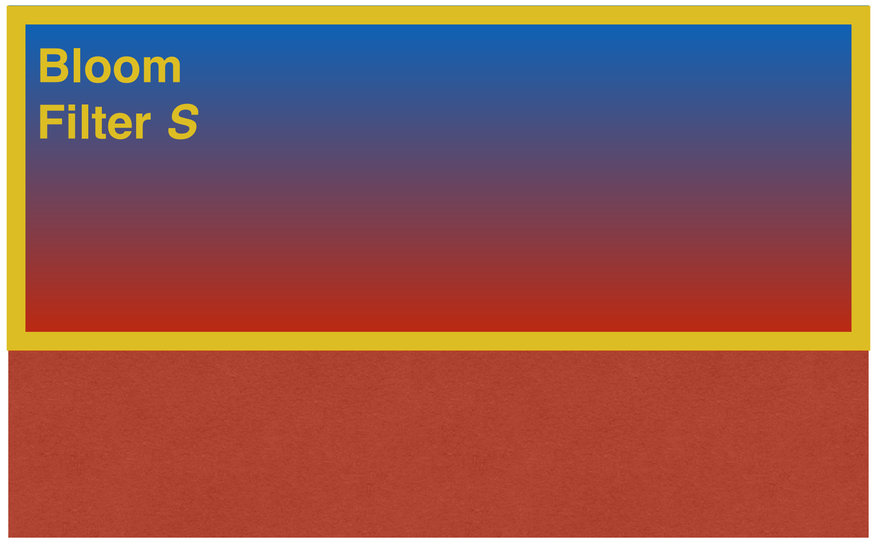
Graphene: Protocol 1

- The block is a subset of the mempool

Sender 

Receiver 



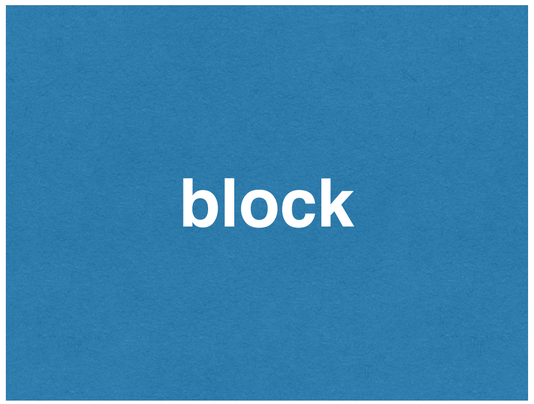


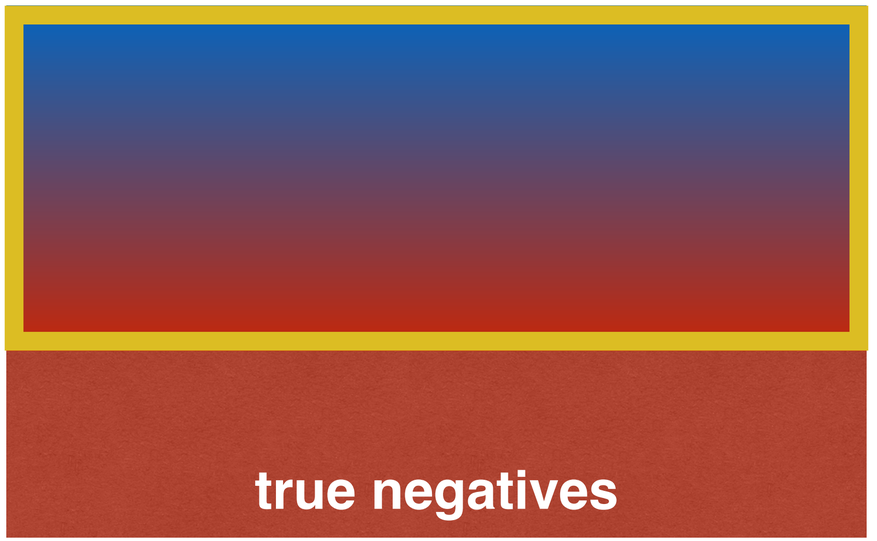
Graphene: Protocol 1

- The block is a subset of the mempool

Sender 

Receiver 



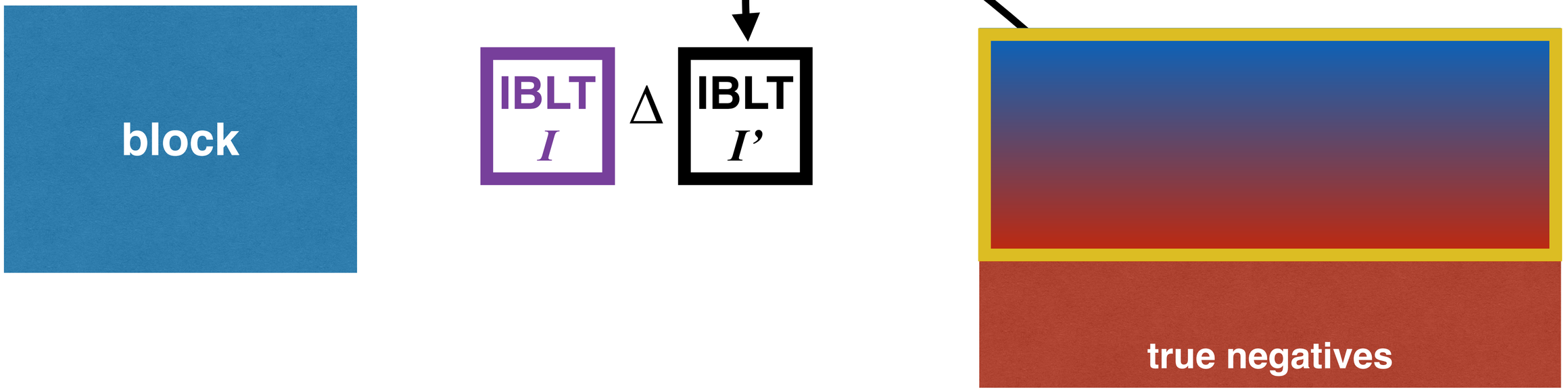


Graphene: Protocol 1

- The block is a subset of the mempool

Sender 

Receiver 

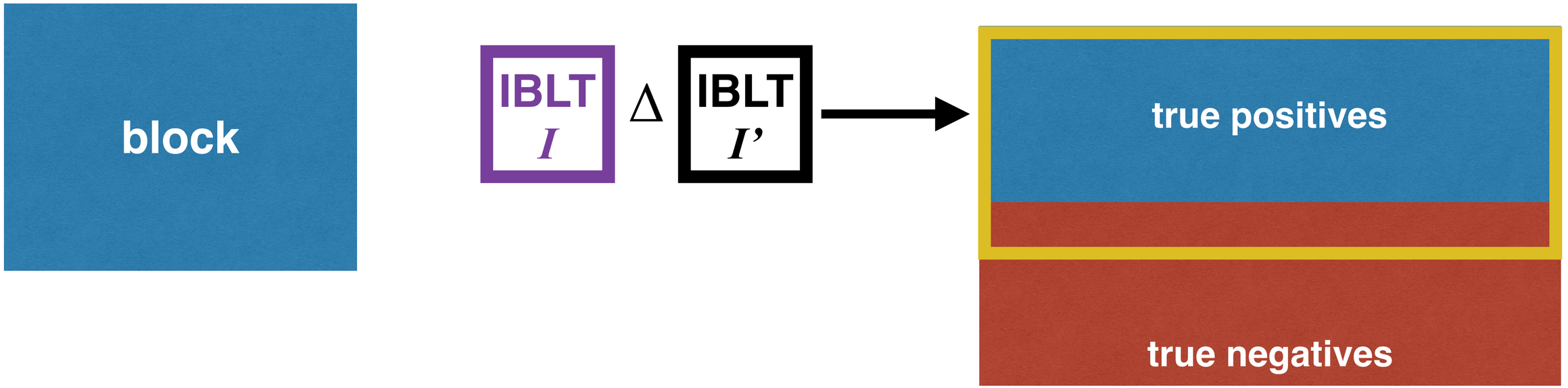


Graphene: Protocol 1

- The block is a subset of the mempool

Sender 

Receiver 

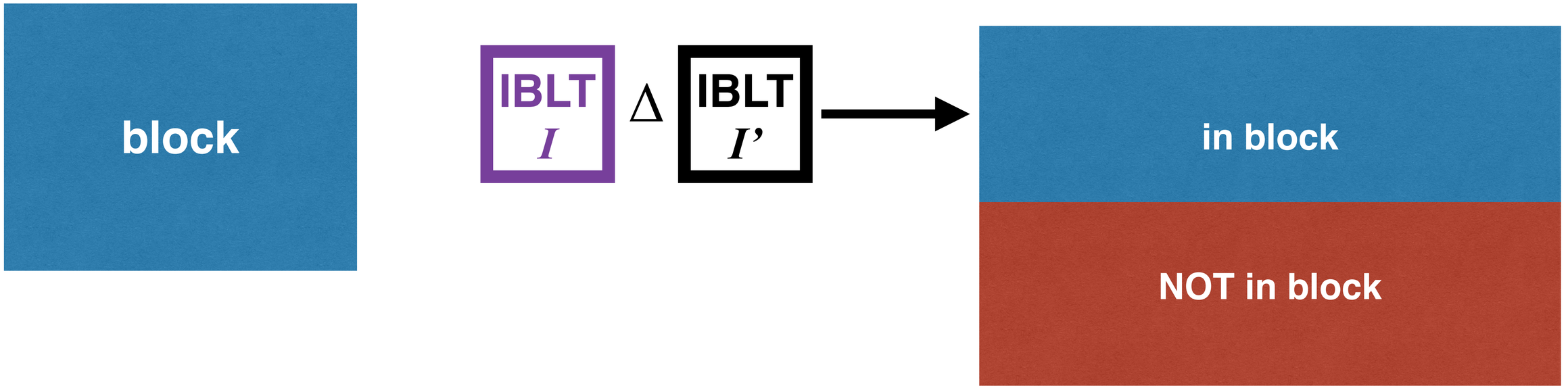


Graphene: Protocol 1

- The block is a subset of the mempool

Sender 

Receiver 

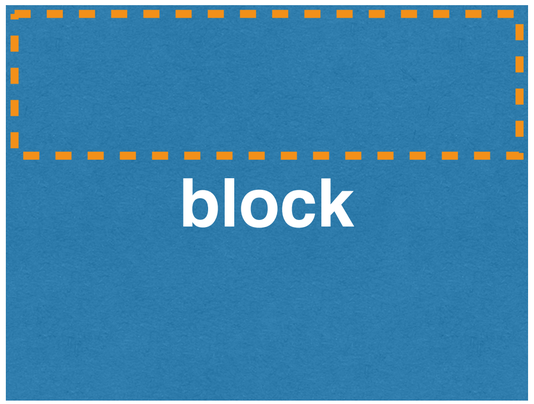


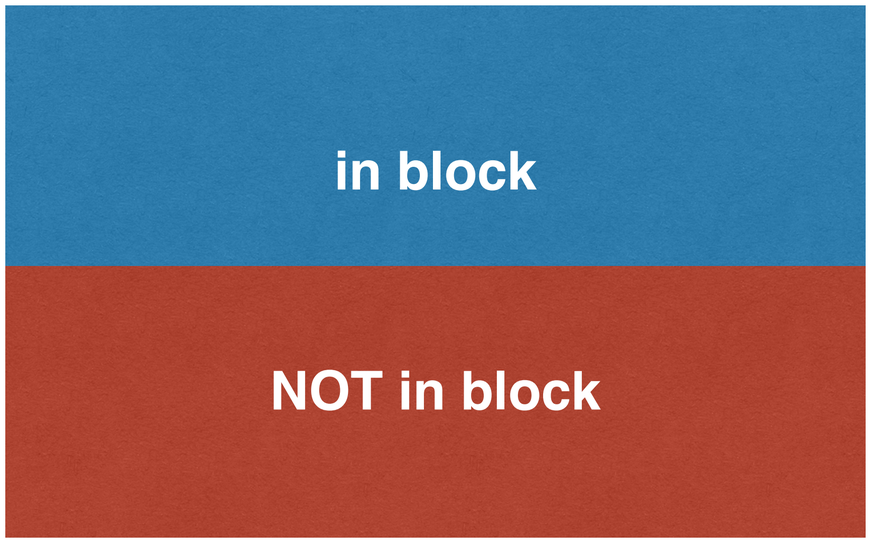
Graphene: Protocol 1

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

Sender 

Receiver 

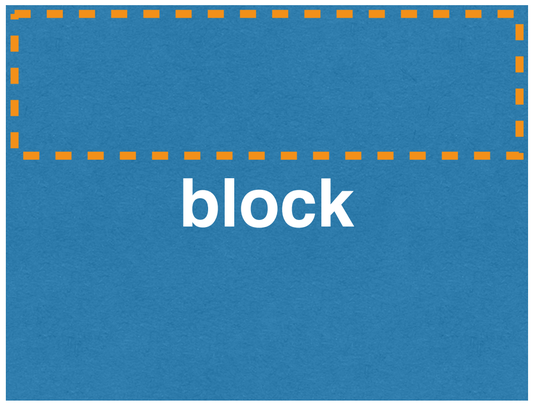




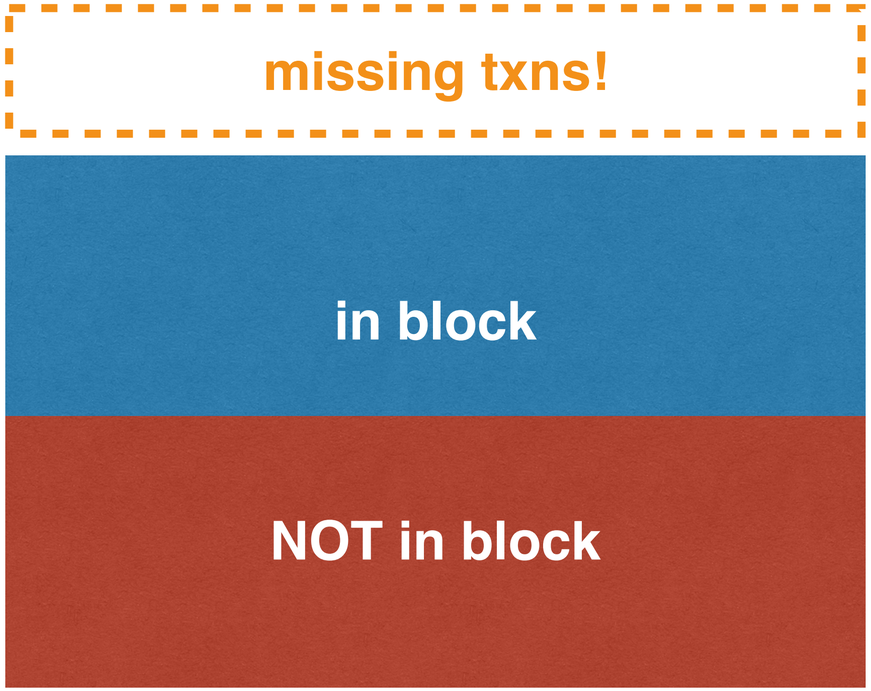
Graphene: Protocol 1

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

Sender 



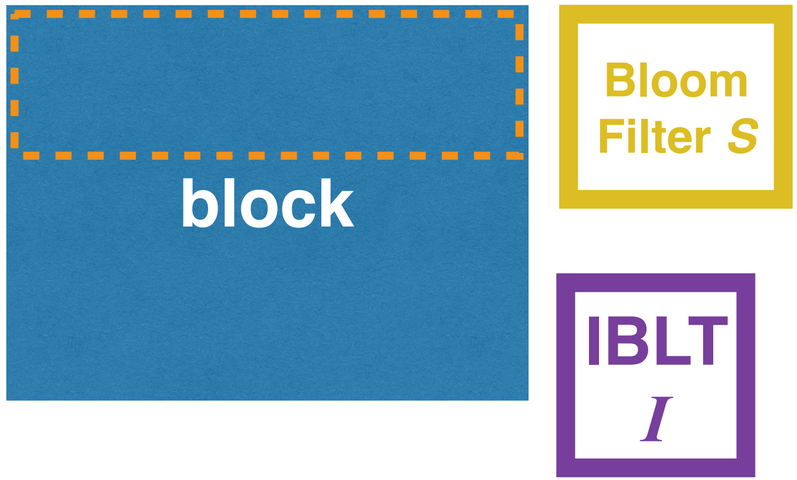
Receiver 



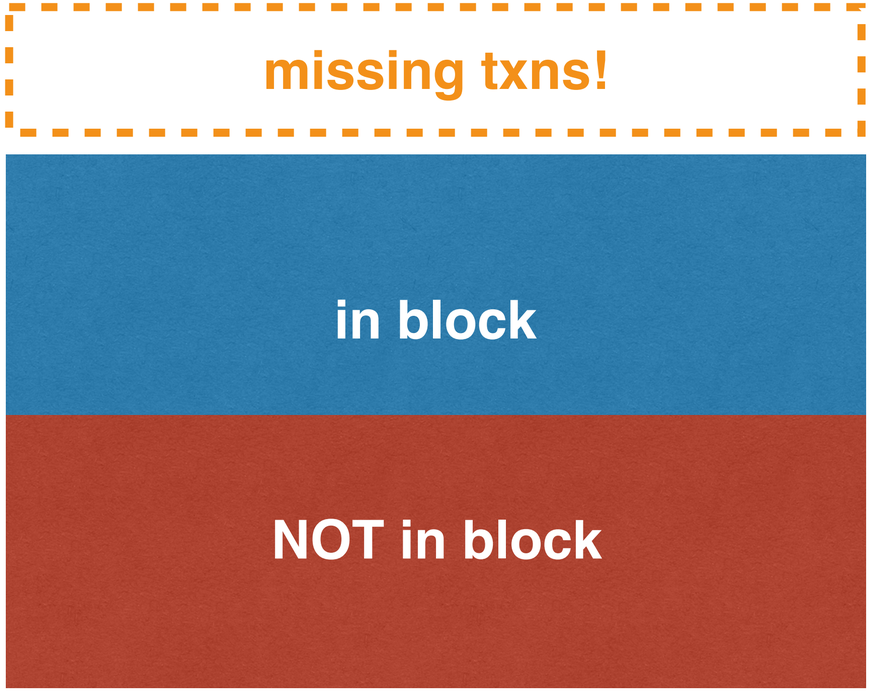
Graphene: Protocol 1

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

Sender 



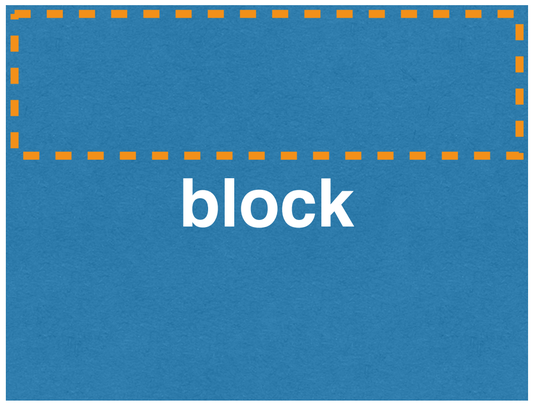
Receiver 



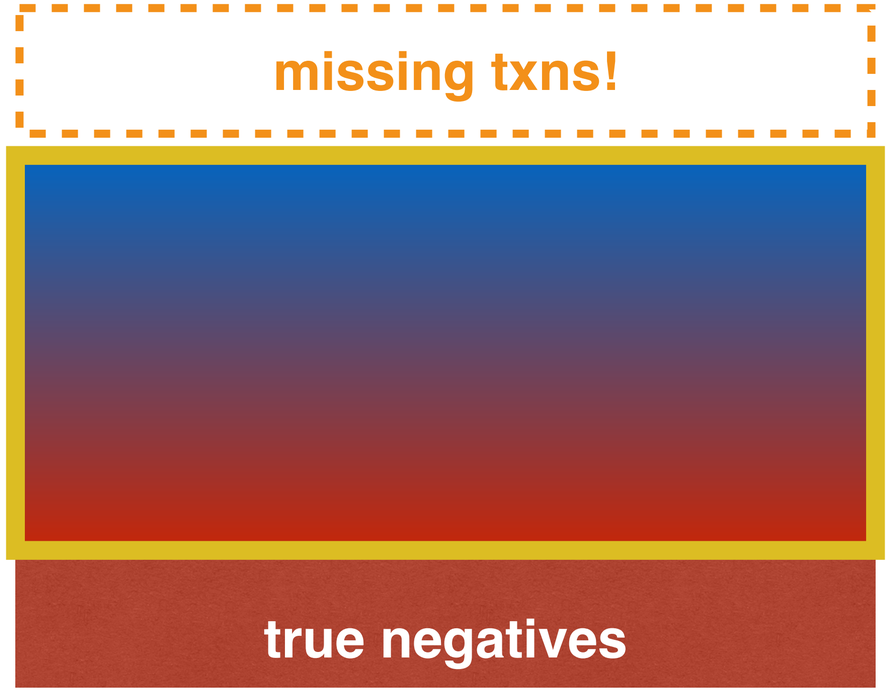
Graphene: Protocol 1

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

Sender 



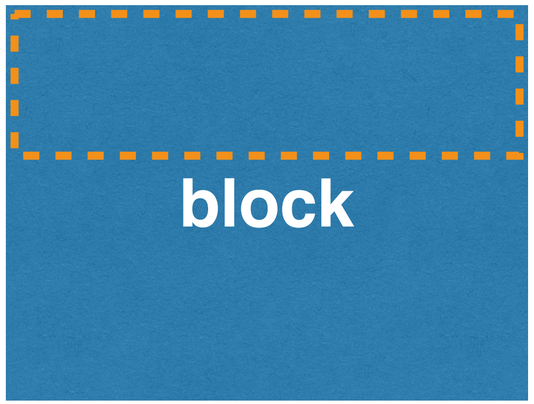
Receiver 

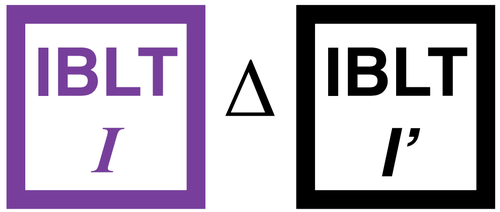


Graphene: Protocol 1

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

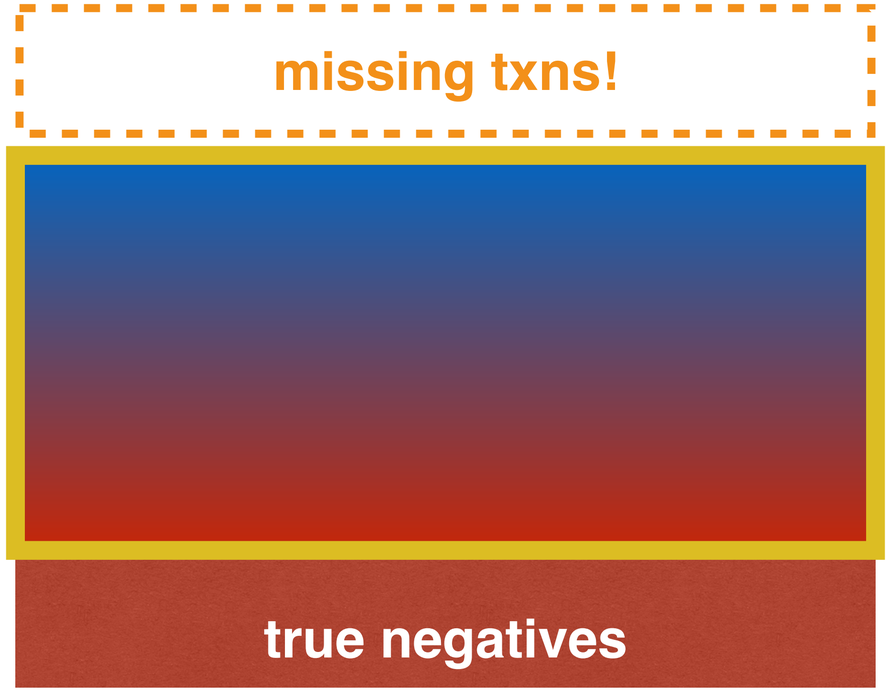
Sender 





Can't decode!

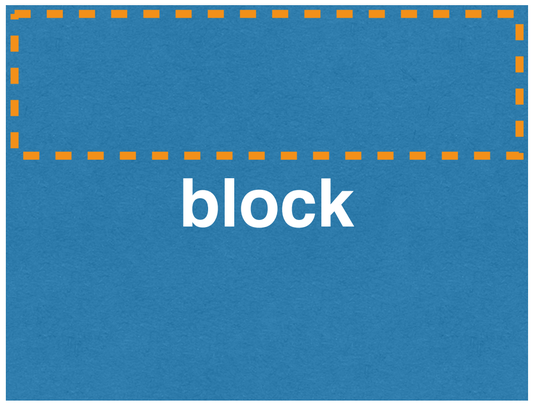
Receiver 



Graphene: Protocol 2

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

Sender 

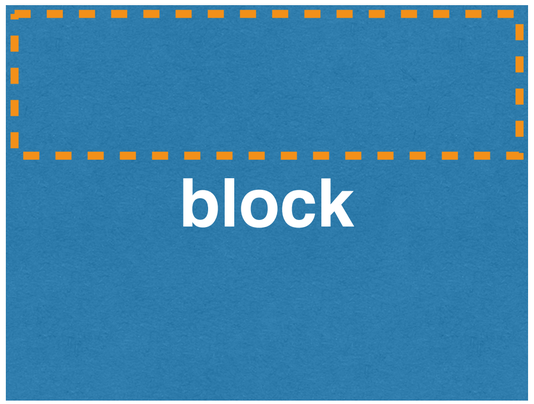


Receiver 

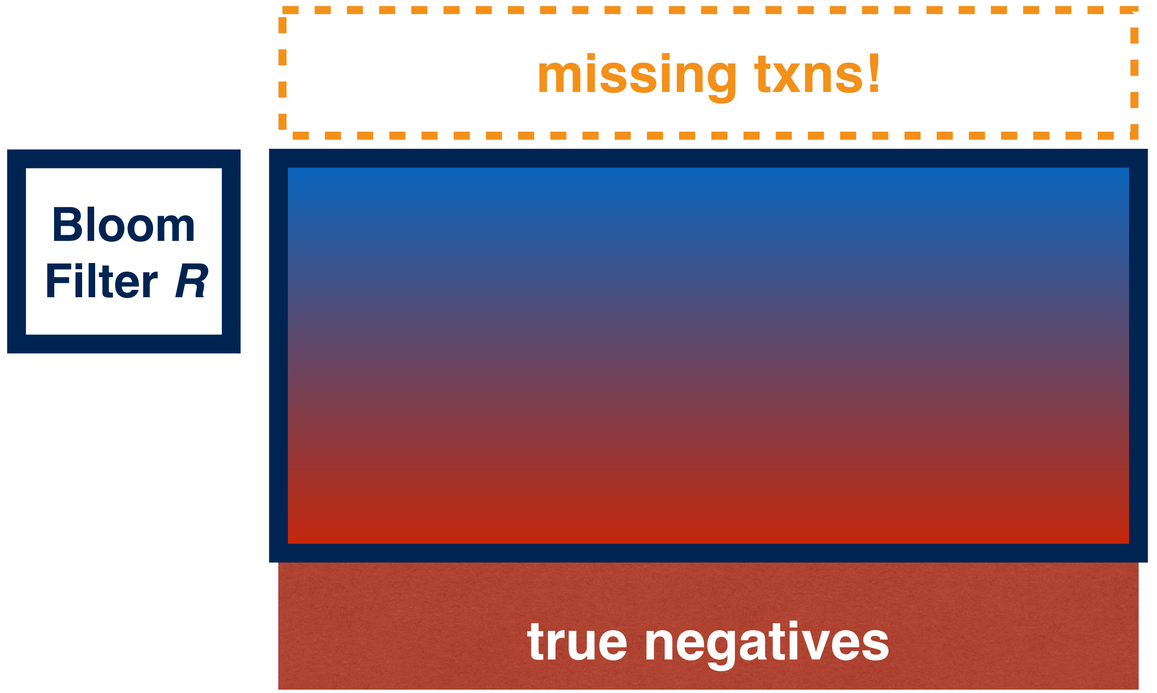
Graphene: Protocol 2

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

Sender 



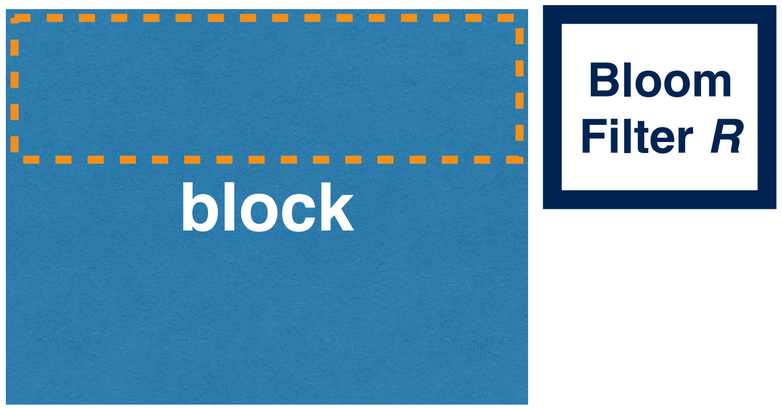
Receiver 



Graphene: Protocol 2

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

Sender 

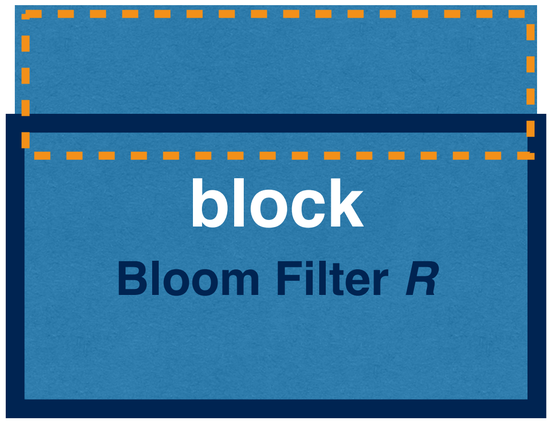


Receiver 

Graphene: Protocol 2

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

Sender 

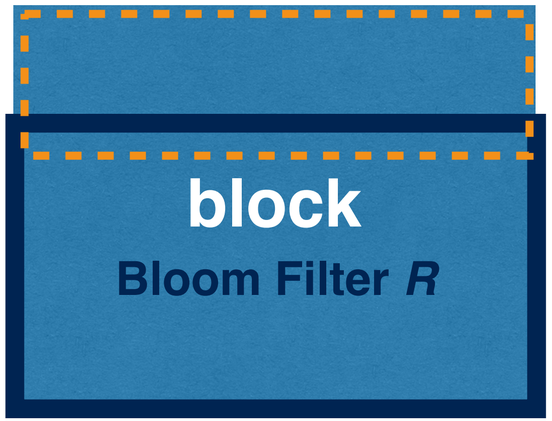


Receiver 

Graphene: Protocol 2

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

Sender 

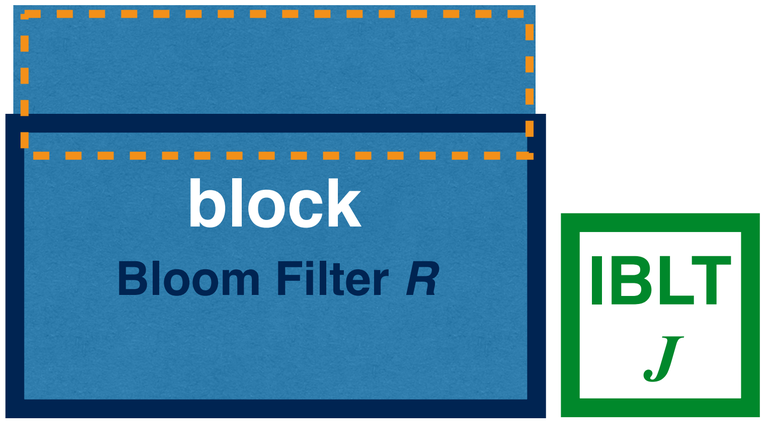


Receiver 

Graphene: Protocol 2

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

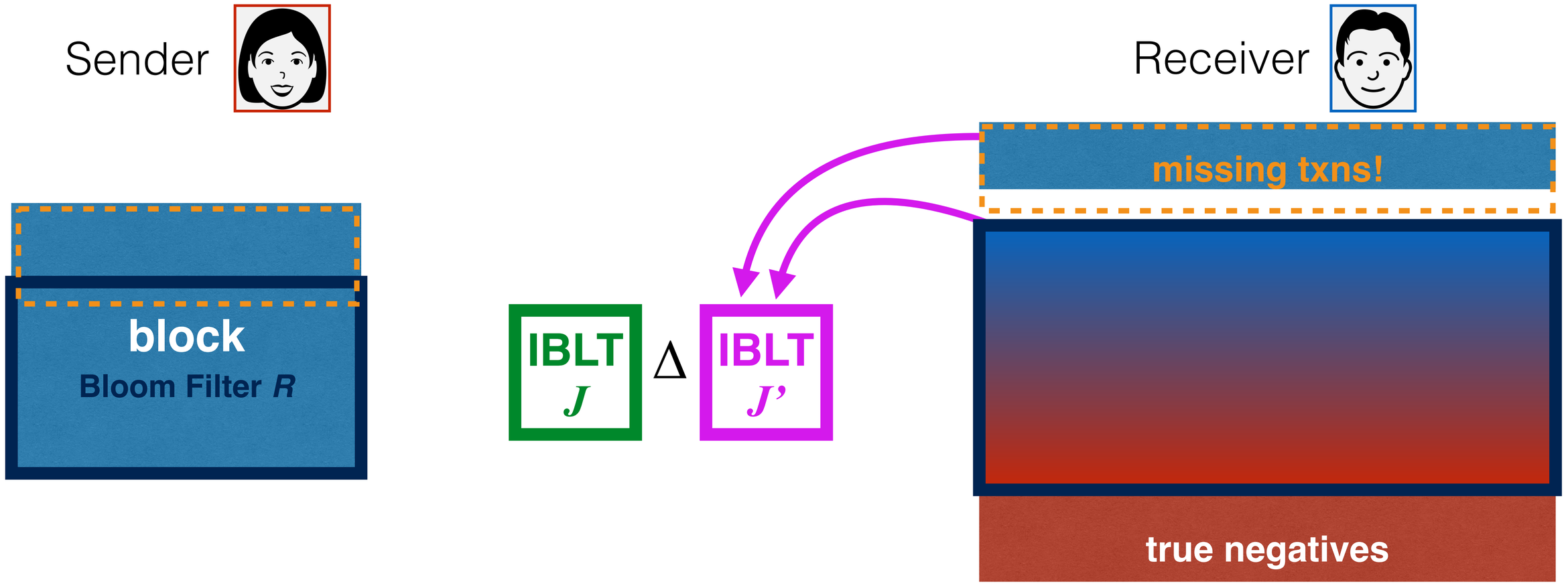
Sender 



Receiver 

Graphene: Protocol 2

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

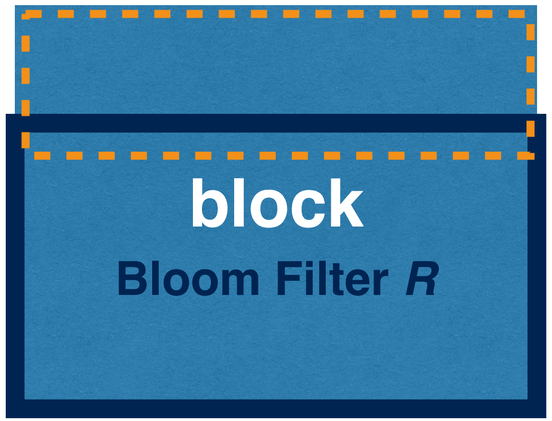


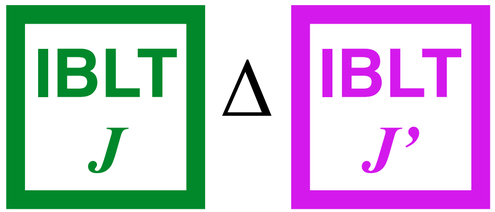
Graphene: Protocol 2

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

Sender 

Receiver 

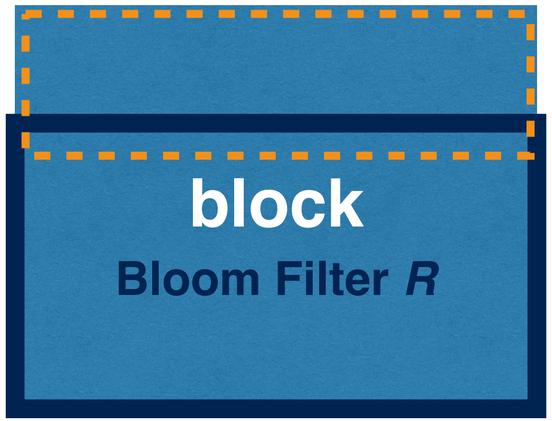


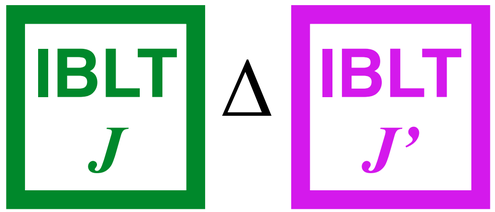


Graphene: Protocol 2

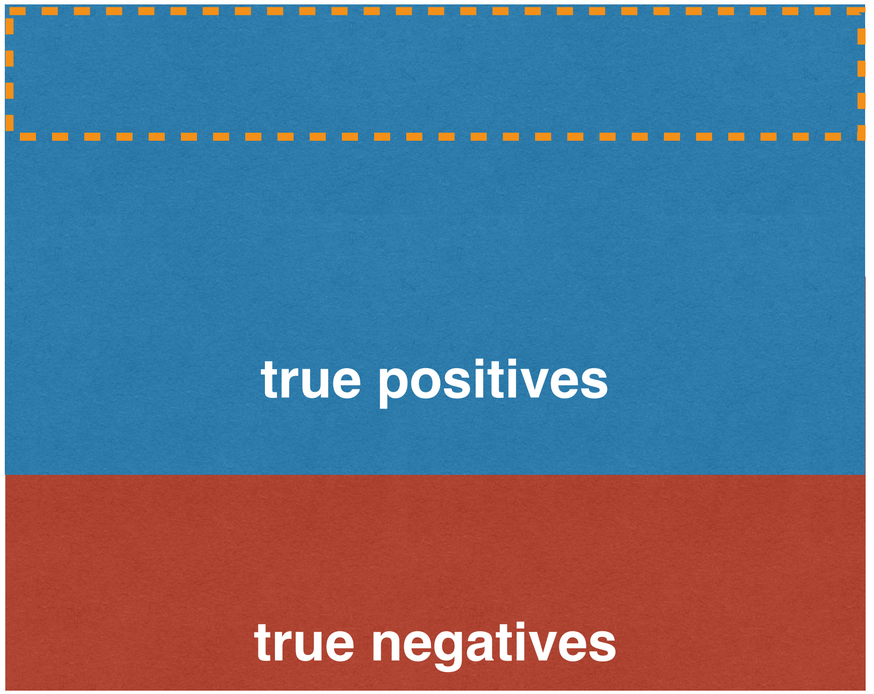
- The block is *not* a subset of the mempool

Sender 





Receiver 

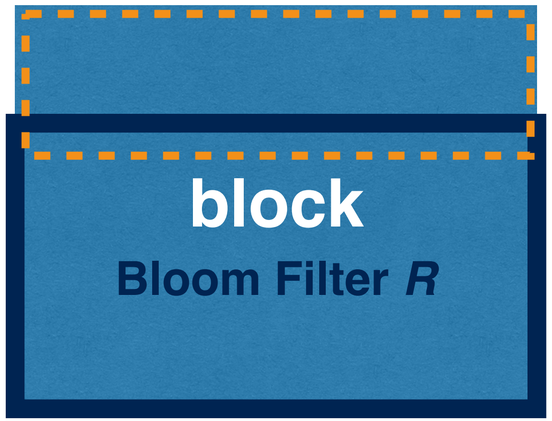


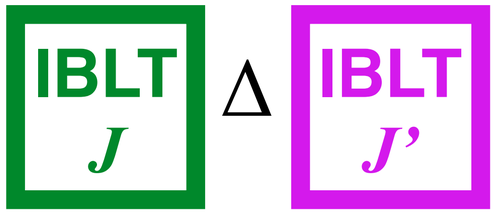
Graphene: Protocol 2

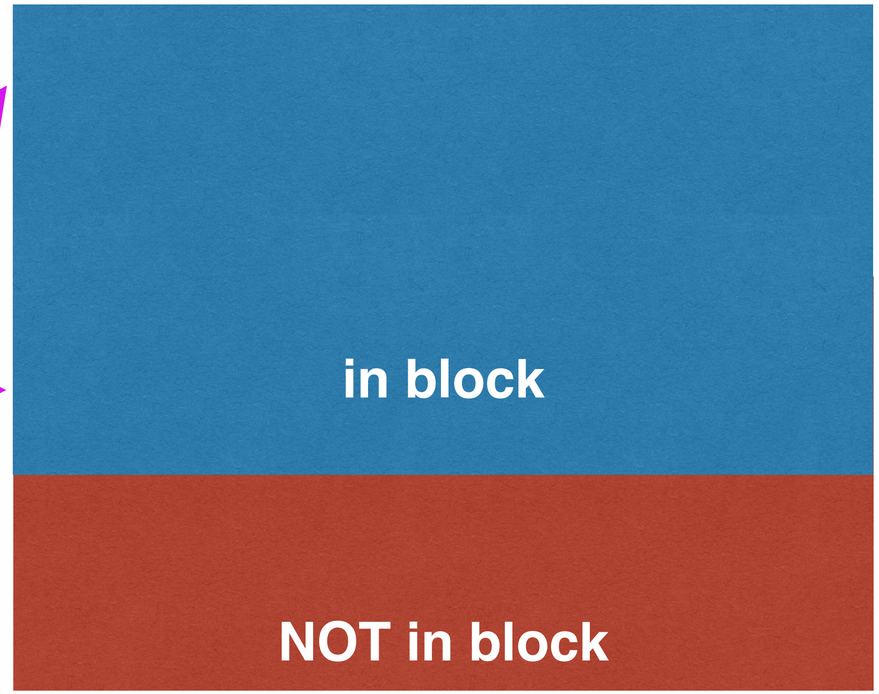
- The block is *not* a subset of the mempool

Sender 

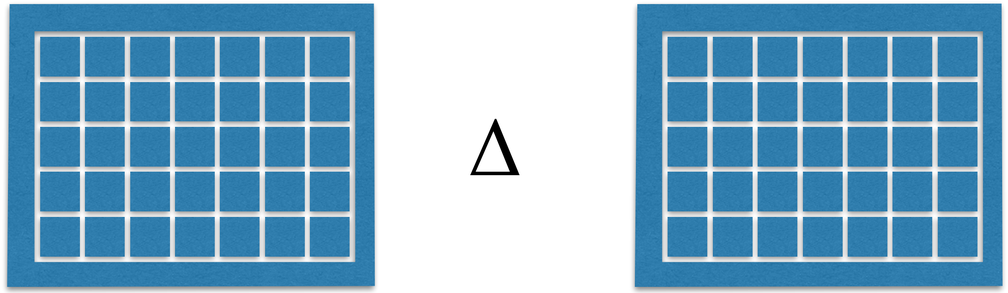
Receiver 







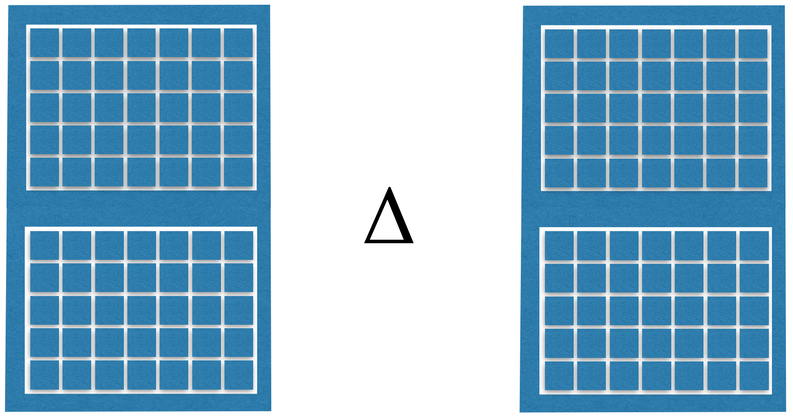
IBLTs



- **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

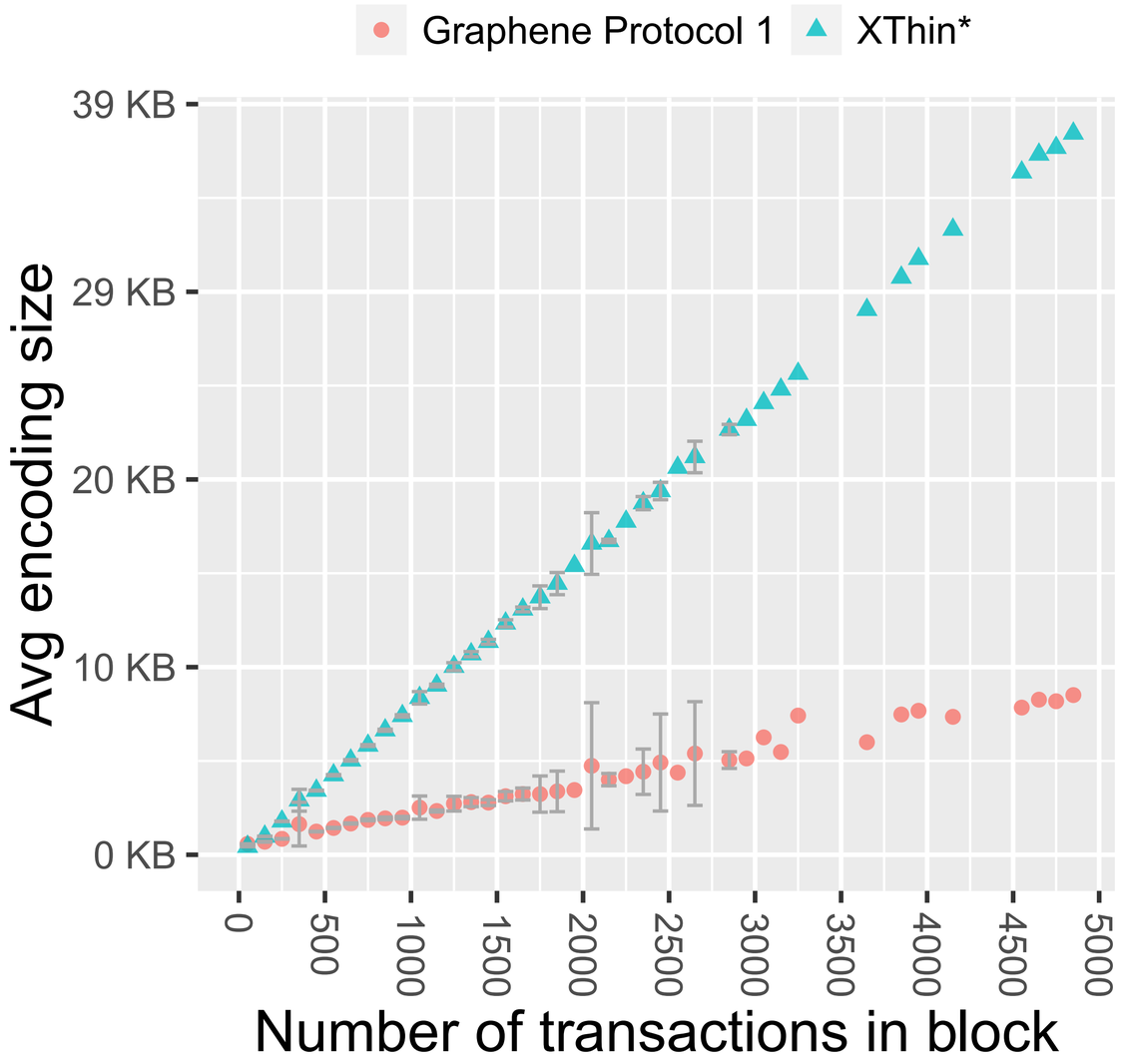
IBLTs



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

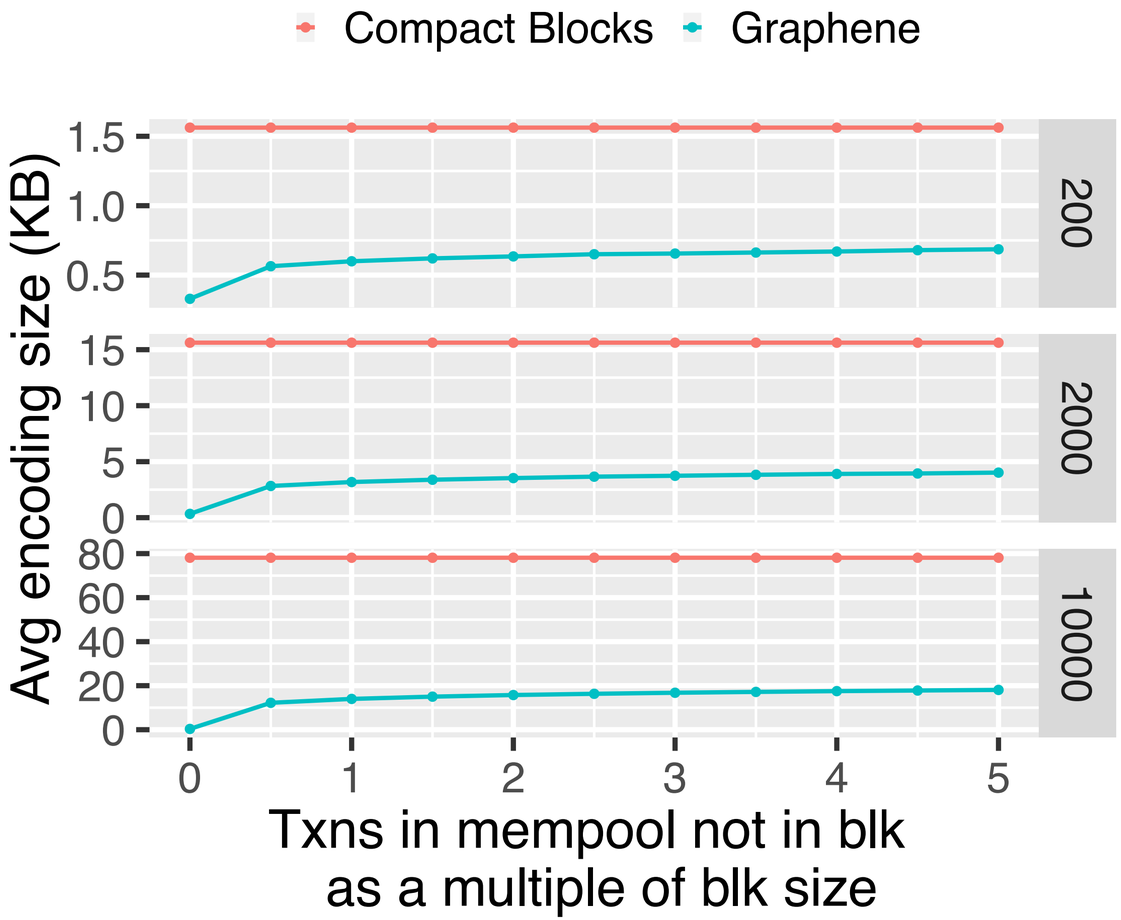
Open-Source Deployment

- 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



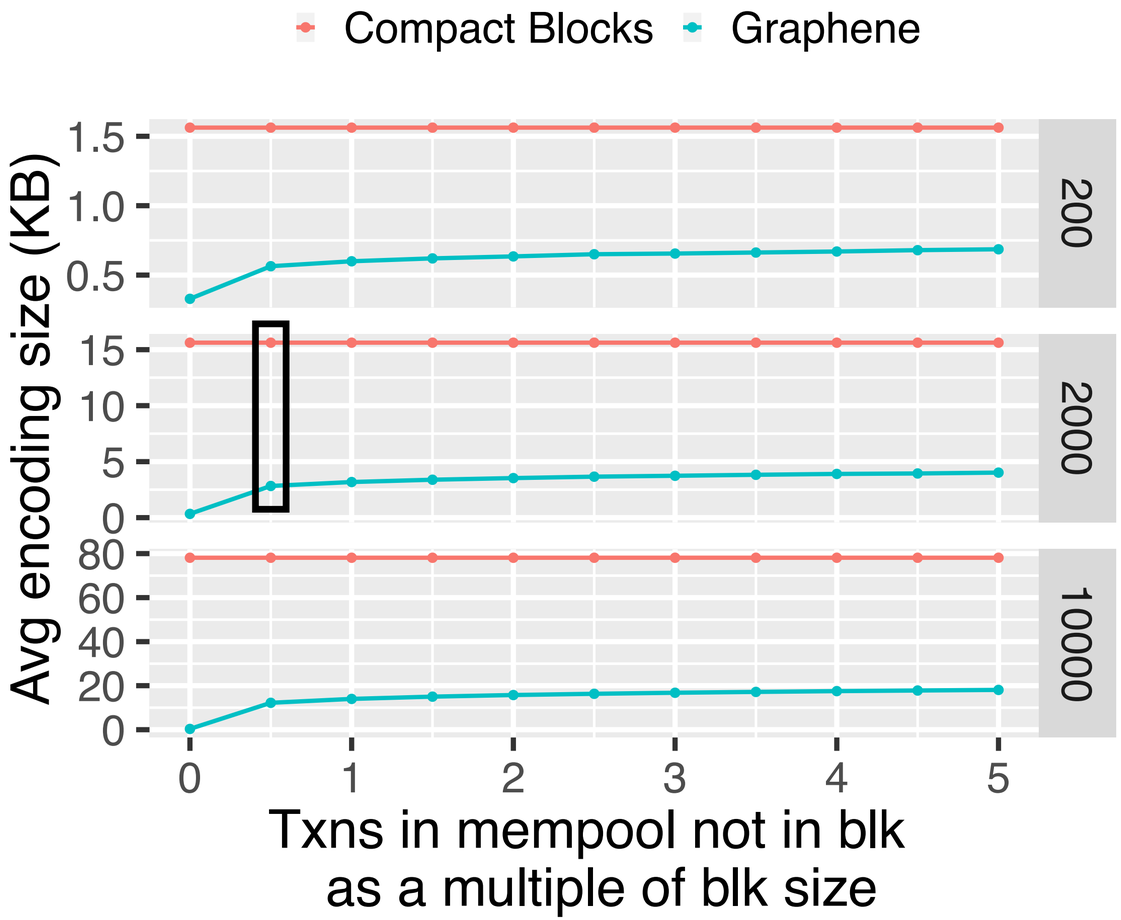
Evaluation

- 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

Thank you!

- **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
  - Snapshot of Graphene code: https://github.com/BitcoinUnlimited/BitcoinUnlimited/blob/release/src/blockrelay/graphene_set.cpp
  - 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