chaincode

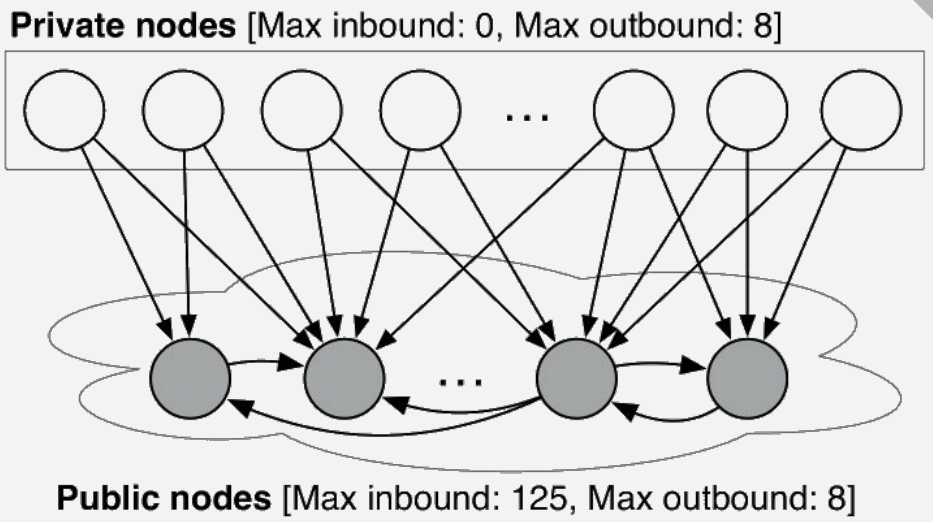
Erlay: Efficient transaction relay

Gleb Naumenko, Greg Maxwell, Pieter Wuille, Sasha Fedorova, Ivan Beschastnikh

Bitcoin p2p network (is redundant)

Ultimately, the network is used to relay transactions.

By default, every node connects to 8 reachable peers.

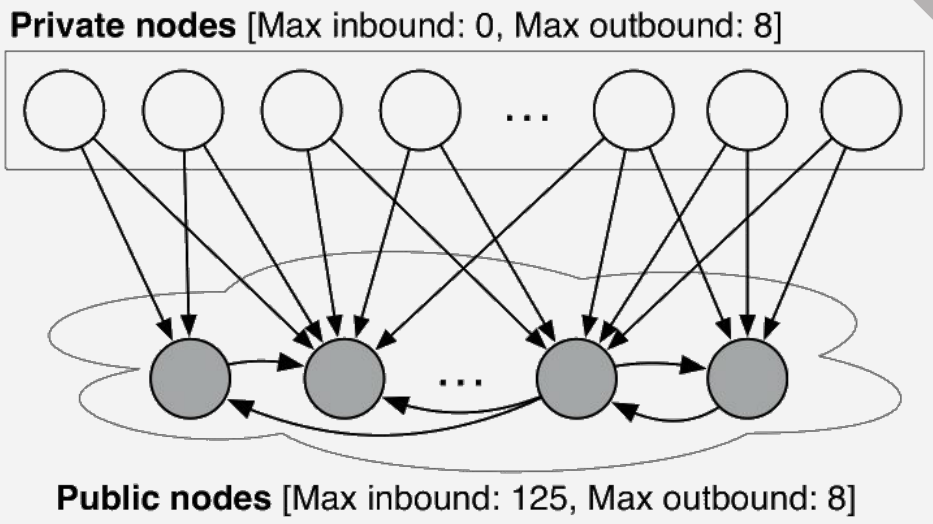


Bitcoin p2p network (is redundant)

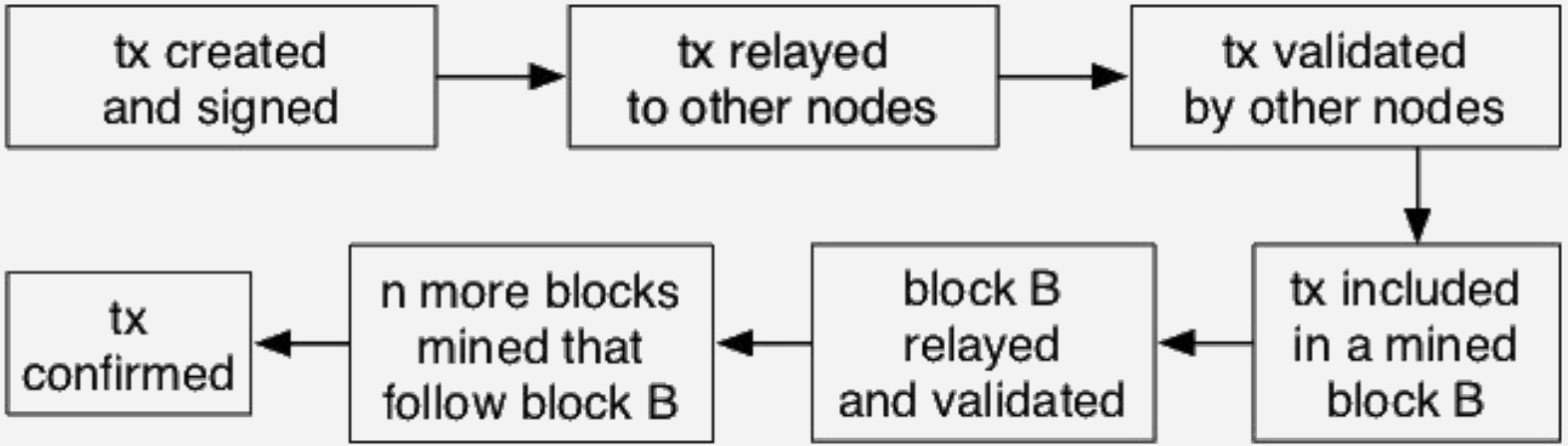
Ultimately, the network is used to relay transactions.

By default, every node connects to 8 reachable peers.

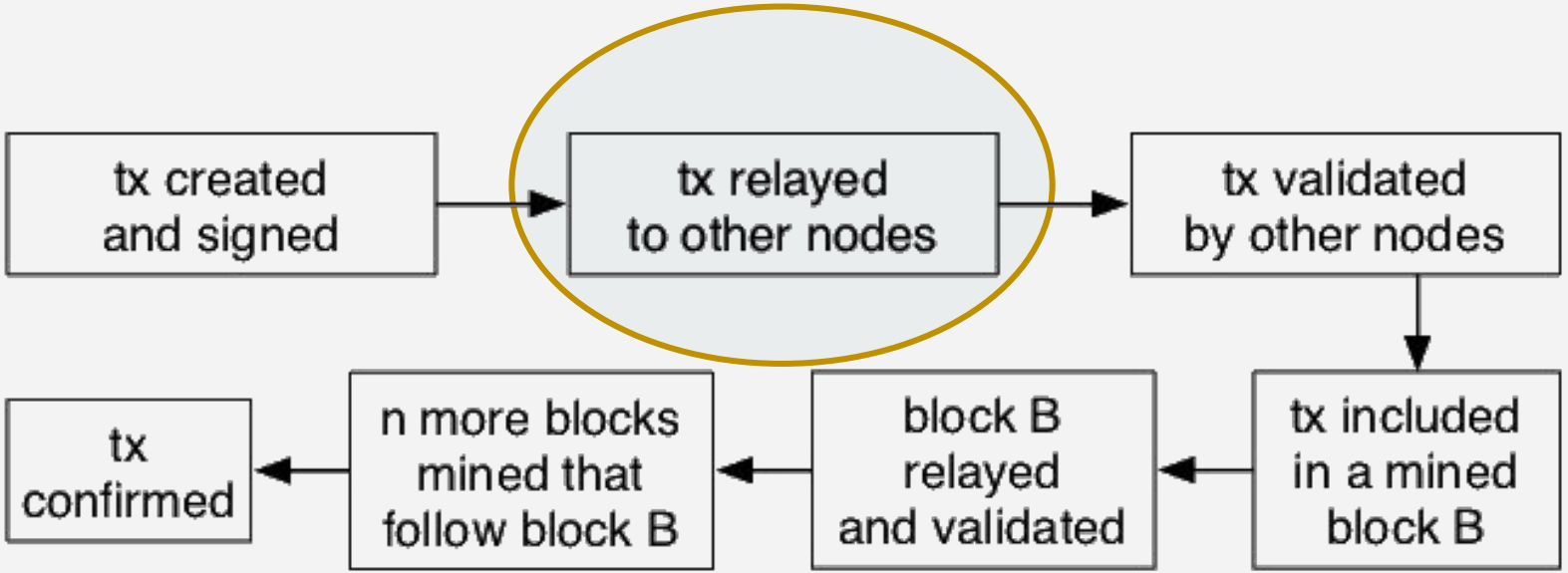
Currently, transactions are relayed by flooding.



Lifecycle of a Bitcoin transaction



Lifecycle of a Bitcoin transaction



Properties of transaction relay

- Bandwidth
- Latency
- Privacy and security

Current protocol (BTCFlood)

- Bandwidth: 18 GB/month
- Latency: 3.15s to reach all nodes
- Privacy and security

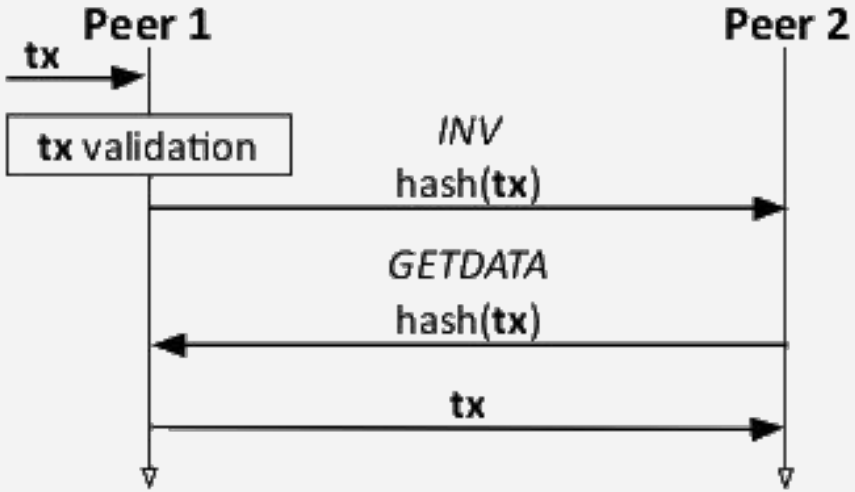
Current protocol (BTCFlood)

- **Bandwidth: 18 GB/month**
- Latency: 3.15s to reach all nodes
- Privacy and security

Current protocol (BTCFlood)

This protocol avoids relaying a full transaction when it is not needed.

- Full transaction is ~250 bytes
- Announcement is 32 bytes

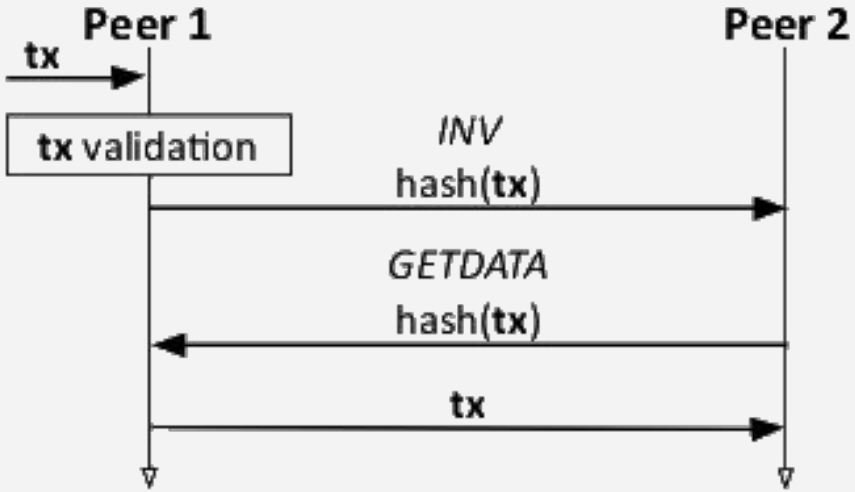


Transaction relay protocol

This protocol avoids relaying a full transaction when it is not needed.

- Full transaction is ~250 bytes
- Announcement is 32 bytes

Announcements are still redundant. More than 85% announcements are "duplicate"



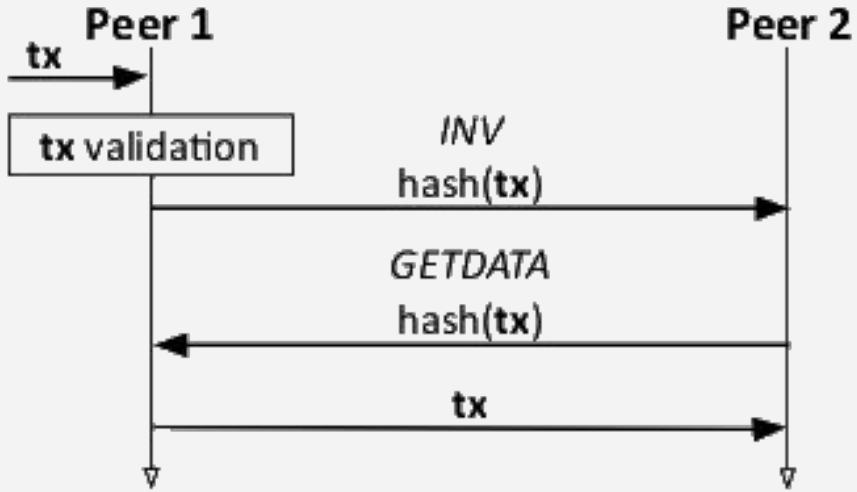
Transaction relay protocol

This protocol avoids relaying a full transaction when it is not needed.

- Full transaction is ~250 bytes
- Announcement is 32 bytes

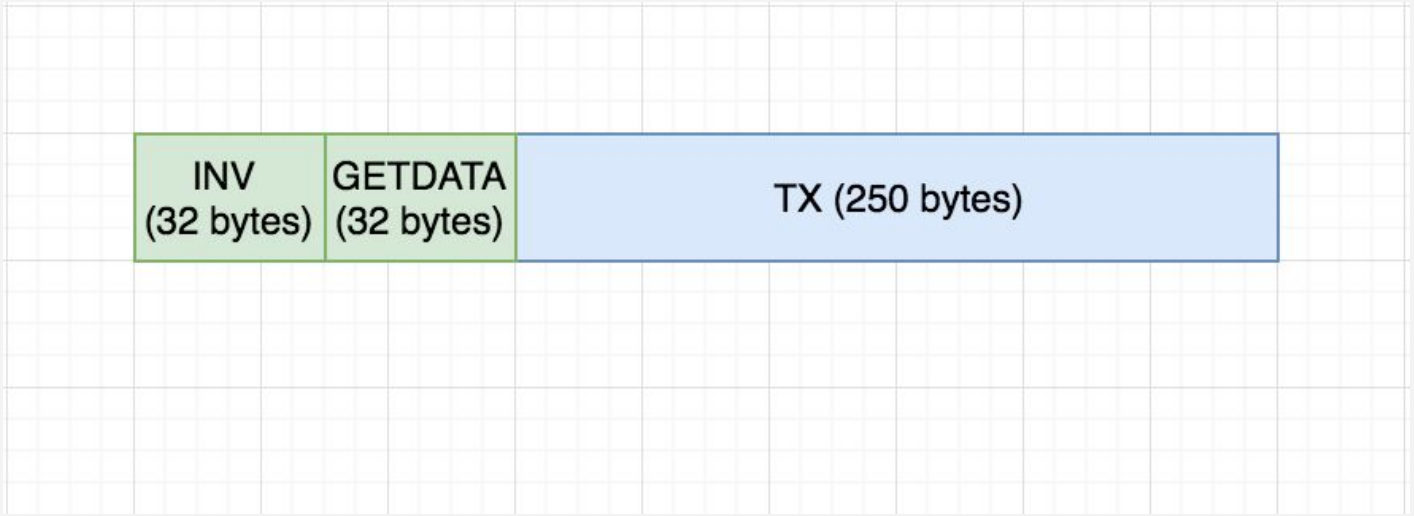
Announcements are still redundant. More than 85% announcements are "duplicate"

Due to the redundancy (8 connections)
Peer 2 may receive this INV 8 times.



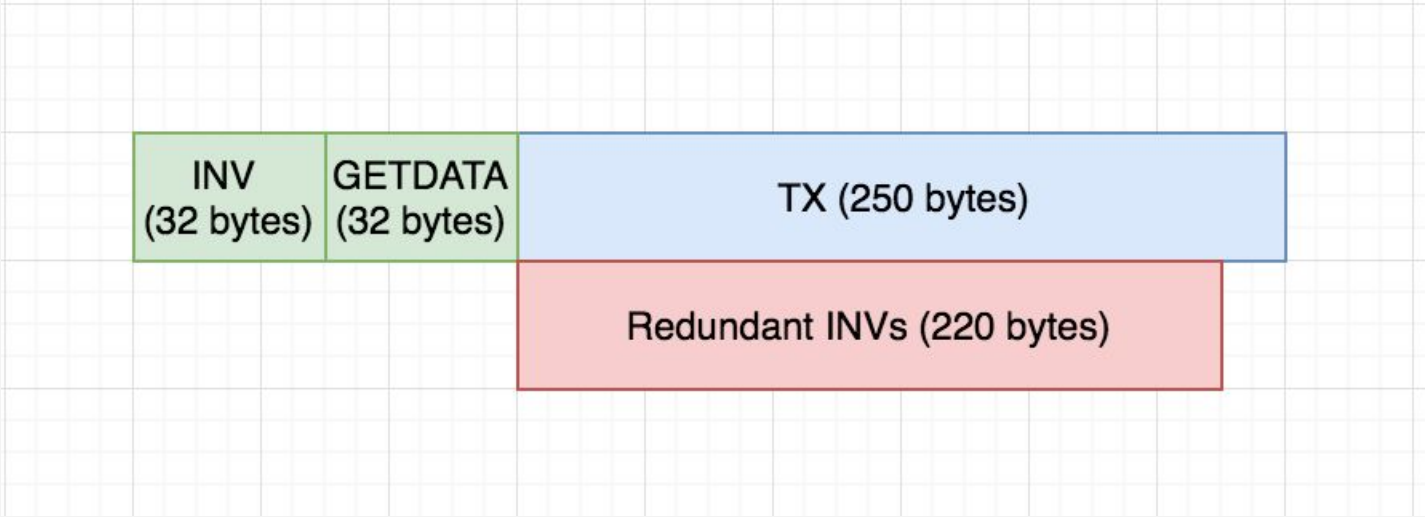
But how bad is that?

Ideally, for every node it is 1 INV, 1 GETDATA and 1 TX message per transaction



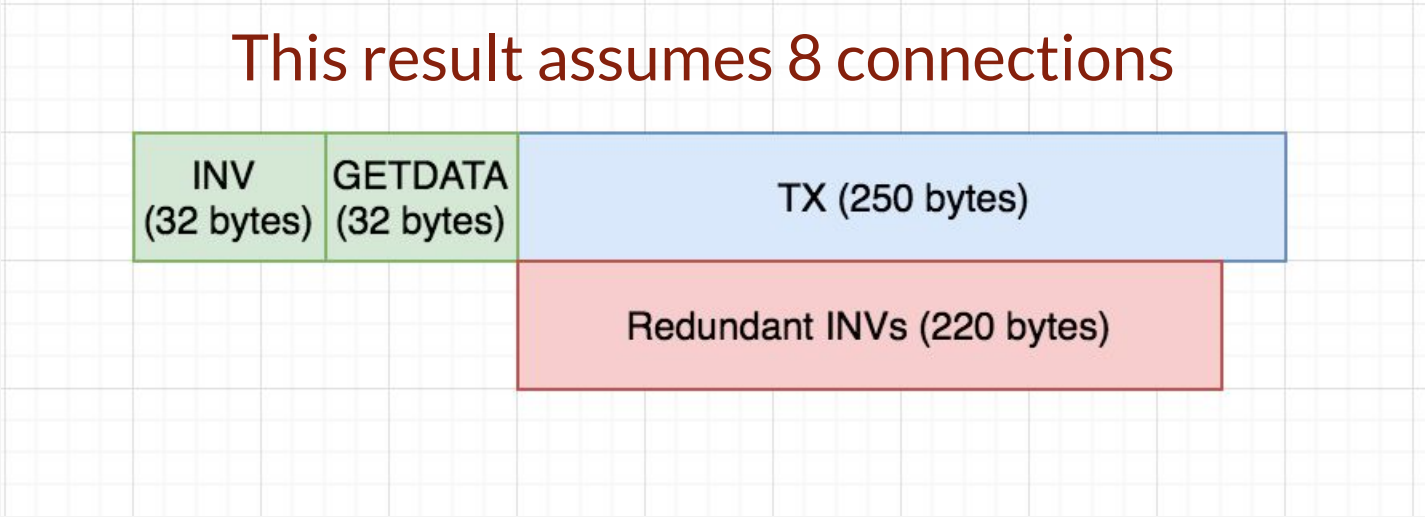
But how bad is that?

Ideally, for every node it is 1 INV, 1 GETDATA and 1 TX message per transaction



But how bad is that?

Ideally, for every node it is 1 INV, 1 GETDATA and 1 TX message per transaction

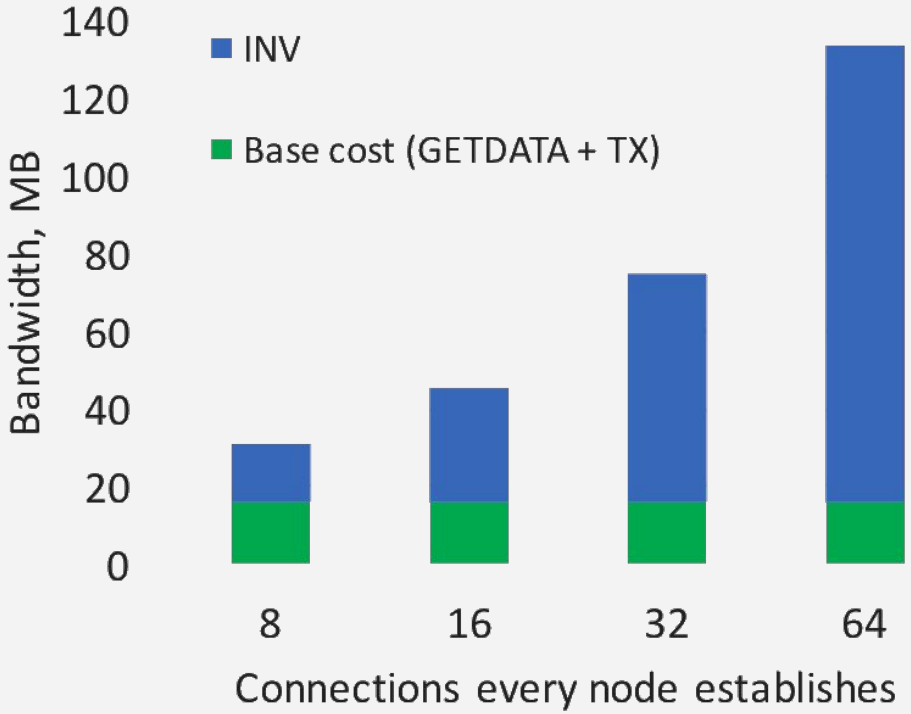


We should increase the connectivity

Makes harder to:

- isolate a node from the network (Eclipsing)
- deanonymize transactions (link tx to the IP-address of the node)
- Infer the topology of the network

But if we increase the connectivity...

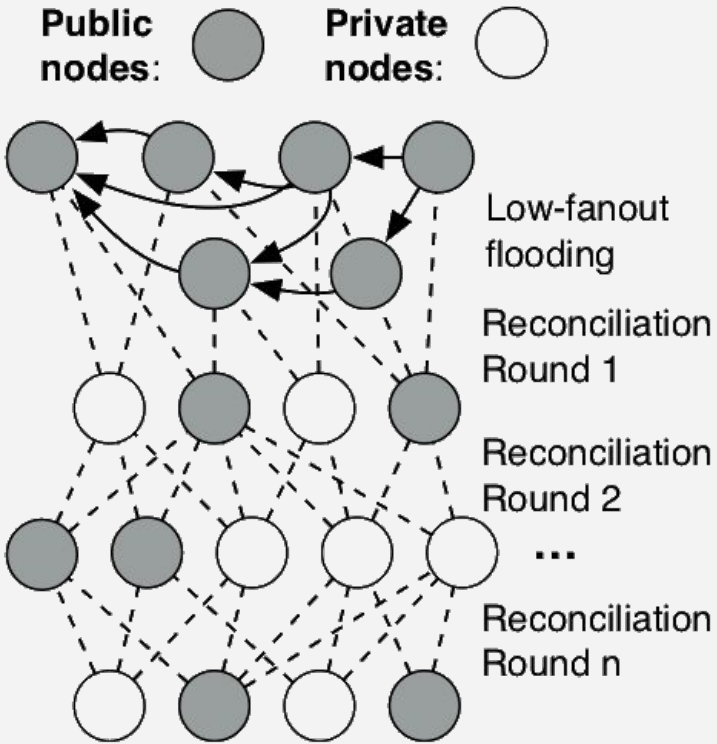


chaincode

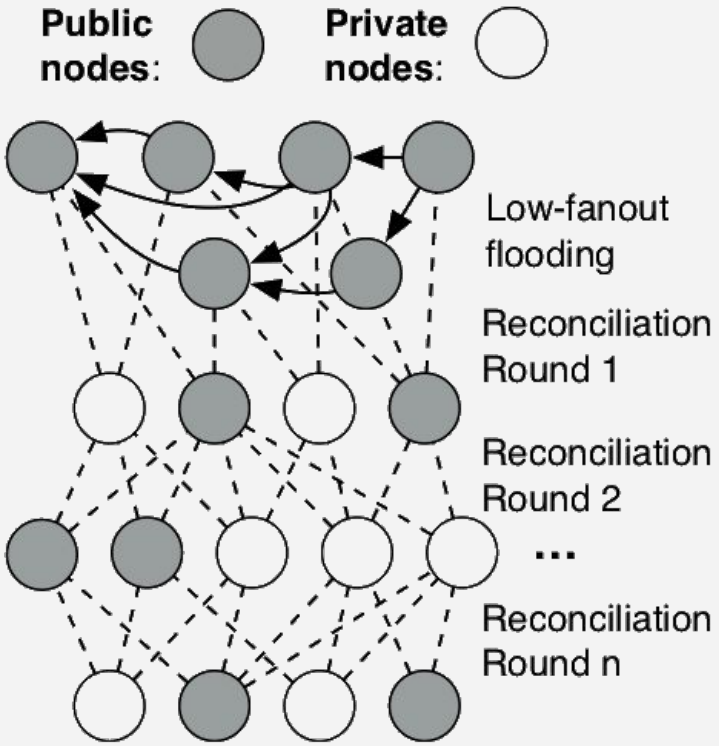
Prior work

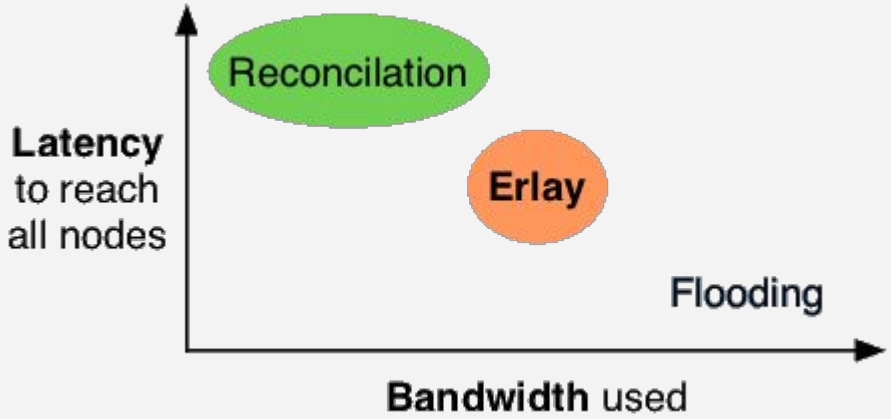
- Block relay protocols: Compact Blocks, Graphene, XThin, bloxroute (different requirements and goals)
- Topology-based routing policies: Freenet, Efa, Chord, Pastry (non-Byzantine)
- Feedback-based approaches (leak information)

Erlay: Efficient transaction relay



Erlay: Efficient transaction relay



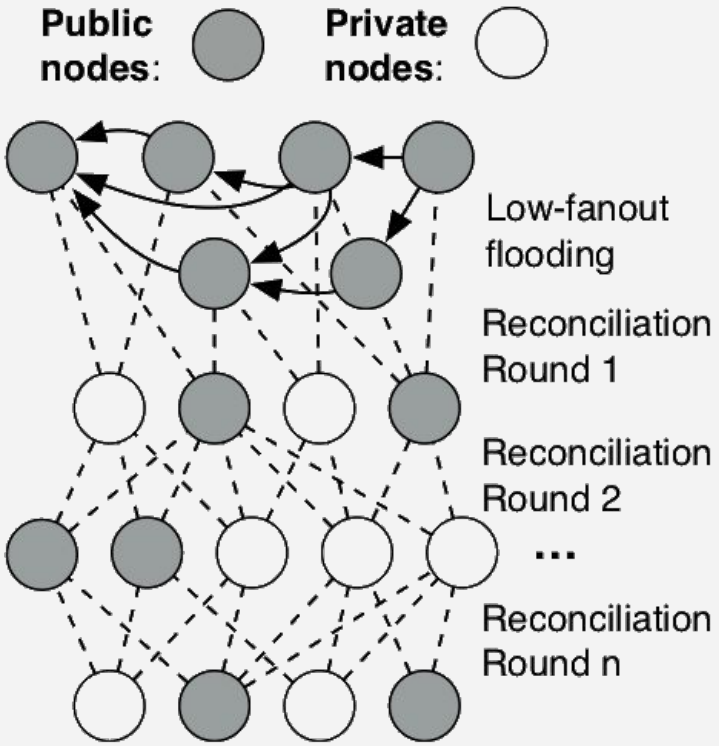


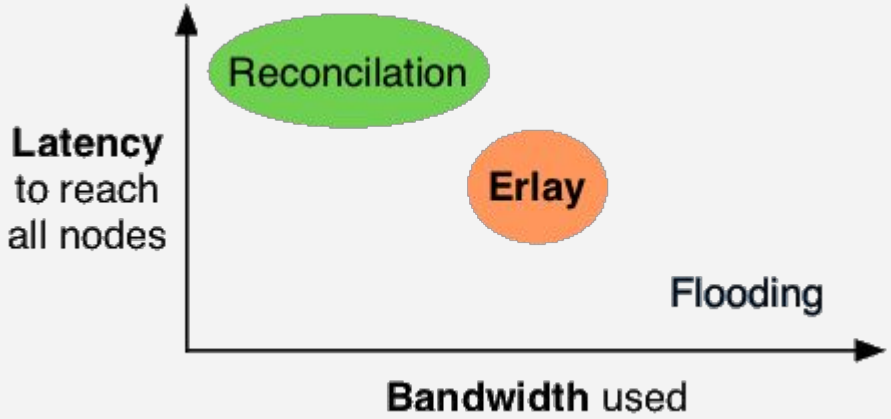
chaincode

Erlay: Efficient transaction relay

1. Rapid flooding across well-connected public (reachable) nodes
2. Every node keeps a reconciliation set for every its peer
3. Include transactions would have been broadcasted through flooding to a given peer in the respective set
4. Use Minisketch for efficient set reconciliation

Erlay: Efficient transaction relay





chaincode

Transaction reconciliation to bridge gaps

The goal of set reconciliation is for each A and B compute

set difference with minimum communication.

Transaction reconciliation to bridge gaps

The goal of set reconciliation is for each A and B compute

set difference with minimum communication.

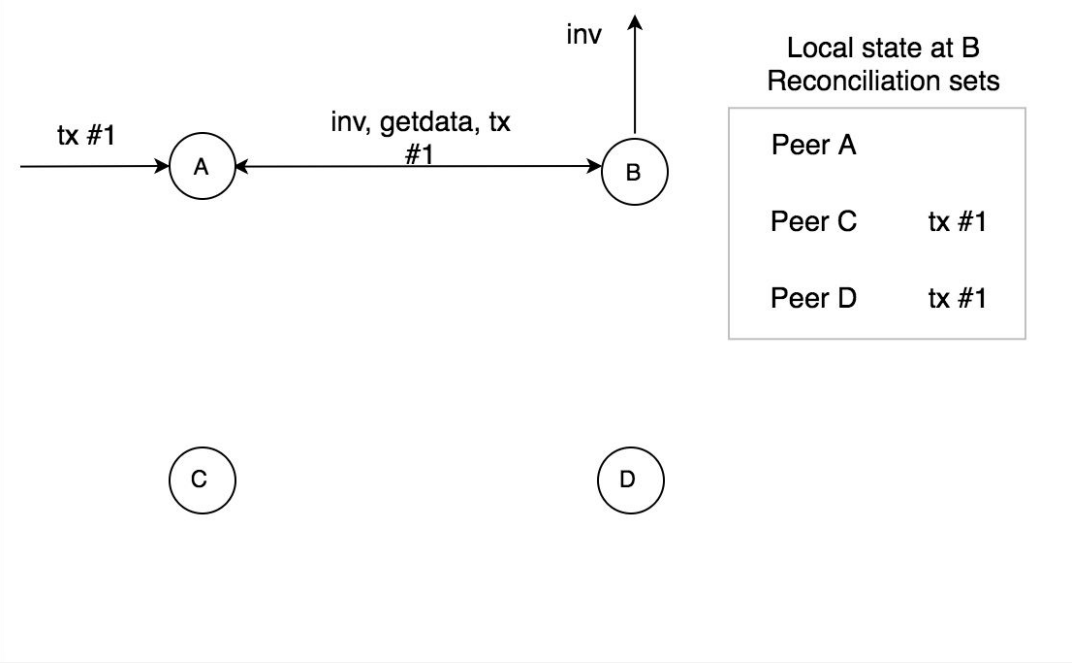
It helps not only to exchange missing transactions, but also to make sure you share the rest of the state.

Transaction reconciliation

Node A knows about tx #1

* A, B, C, D are connected

Transaction reconciliation

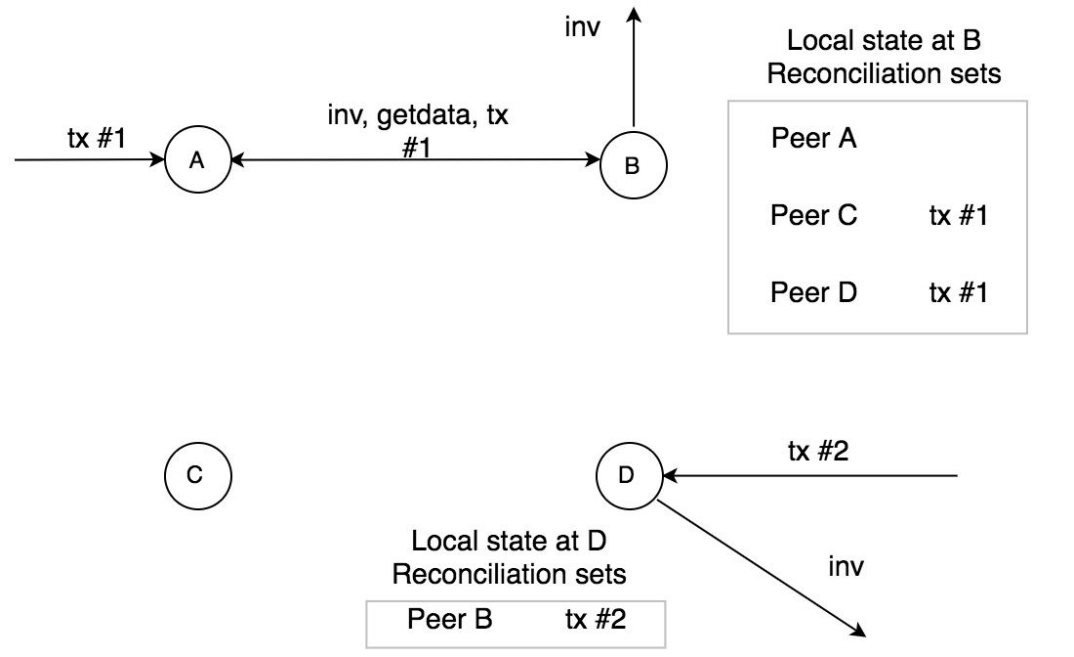


Every node maintains a set of transactions it would have sent to every peer

Nodes A, B know about tx #1

* A, B, C, D are connected

Transaction reconciliation

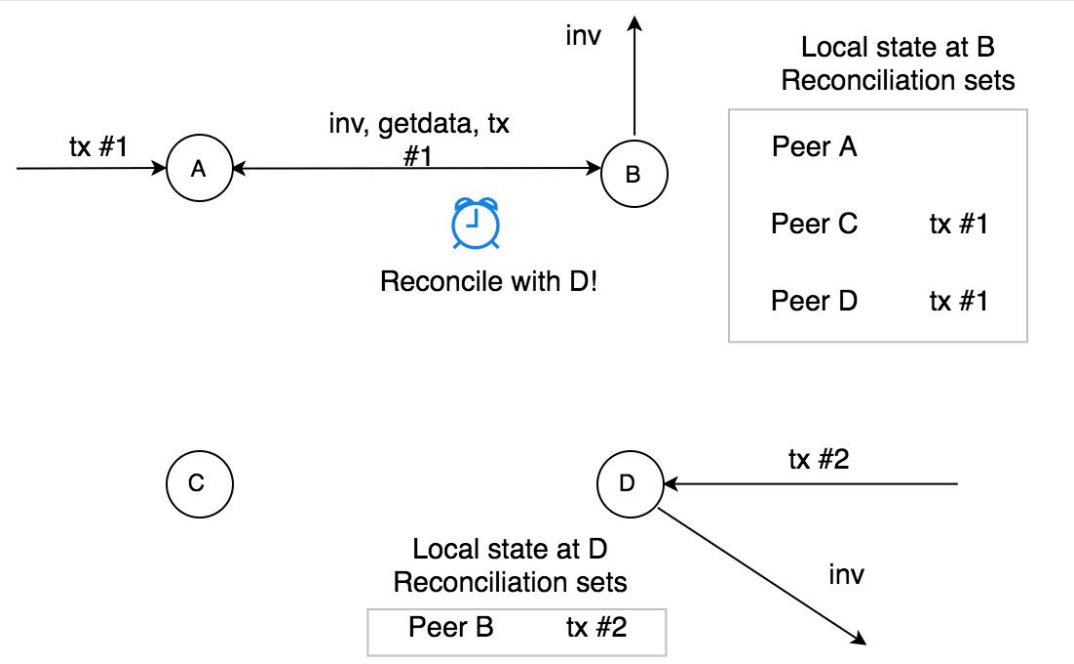


Every node maintains a set of transactions it would have sent to every peer

Nodes A, B know about tx #1
Node D knows about tx #2

* A, B, C, D are connected

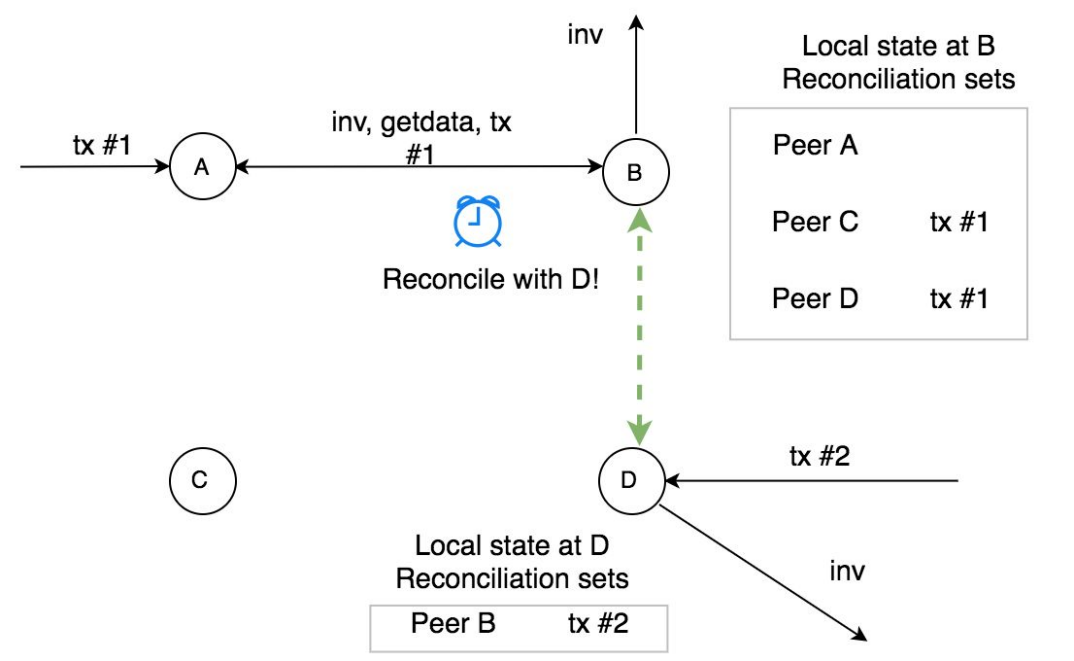
Transaction reconciliation



Nodes A, B know about tx #1
Node D knows about tx #2

* A, B, C, D are connected

Transaction reconciliation

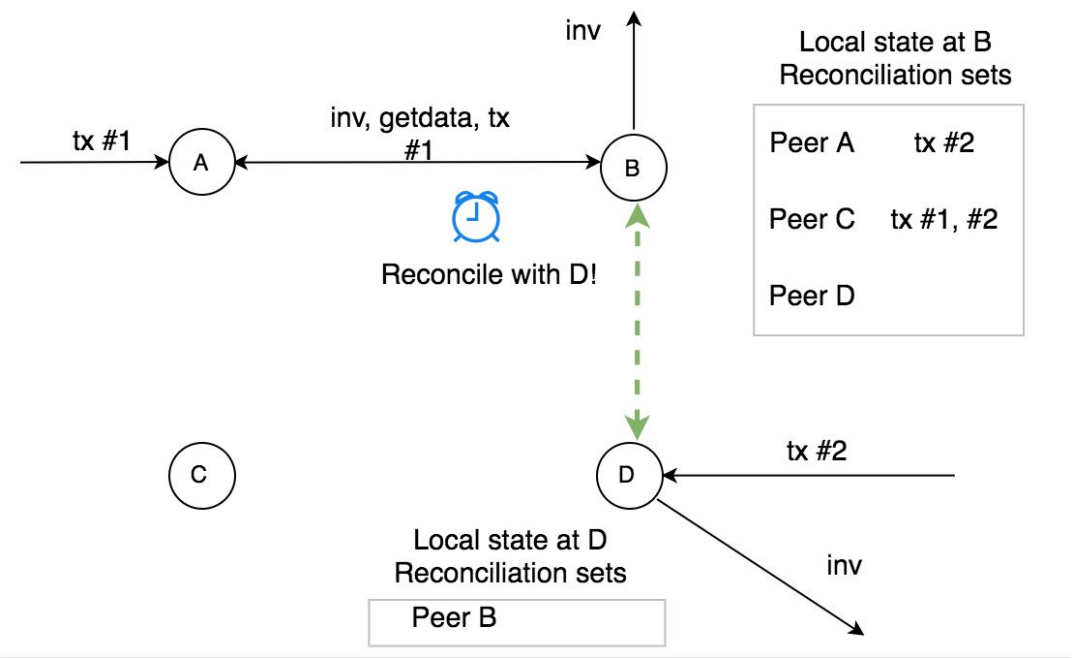


During reconciliation, 2 nodes synchronize according to their local sets

Nodes A, B know about tx #1
Node D knows about tx #2

* A, B, C, D are connected

Transaction reconciliation



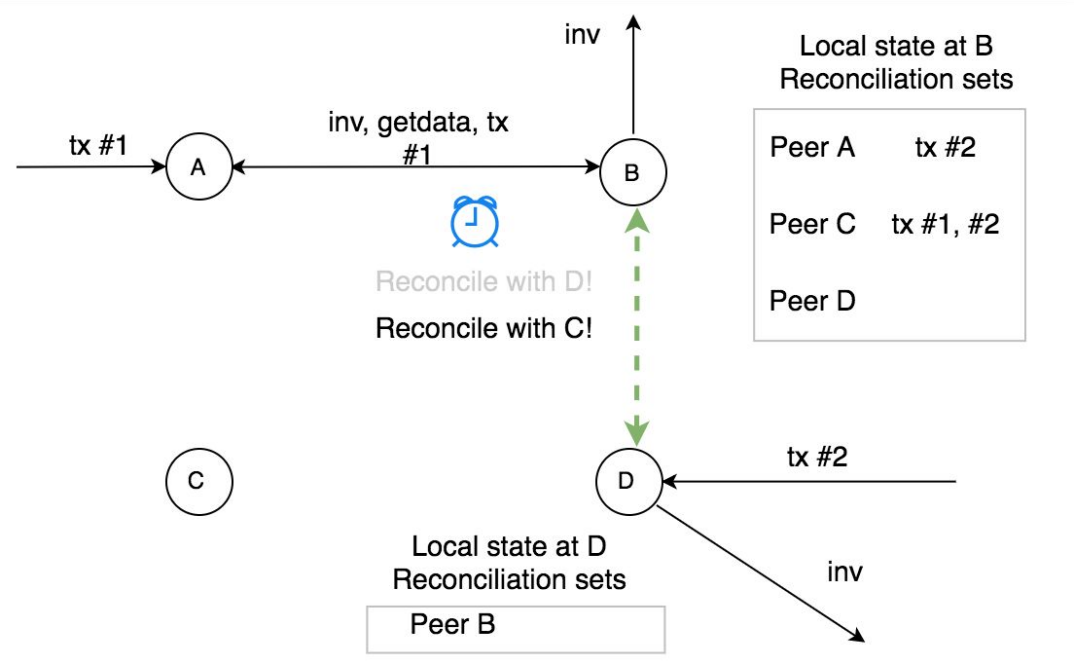
During reconciliation, 2 nodes synchronize according to their local sets.

After reconciliation those sets are cleared.

Nodes A, B, D know about tx #1
Node B, D knows about tx #2

* A, B, C, D are connected

Transaction reconciliation



* A, B, C, D are connected

During reconciliation, 2 nodes synchronize according to their local sets.

After reconciliation those sets are cleared.

Nodes A, B, D know about tx #1
Node B, D knows about tx #2

Finding set difference with BCH codes

Alice	Bob
Tx #1	Tx #1
Tx #2	Tx #2
Tx #3	Tx #3
Tx #4	Tx #4
Tx #5	Tx #5
Tx #6	Tx #6
Tx #10	Tx #15

Bob and Alice never communicated before, but they think they share most of the transactions and want to help each other

Finding set difference with BCH codes

Alice	Bob
Tx #1	Tx #1
Tx #2	Tx #2
Tx #3	Tx #3
Tx #4	Tx #4
Tx #5	Tx #5
Tx #6	Tx #6
Tx #10	Tx #15

Bob and Alice never communicated before, but they think they share most of the transactions and want to help each other

*It turns out that If a transaction ID is 32 bytes, we have to send just 32*2 bytes to find the difference in these sets*

Finding set difference with BCH codes

Alice	Bob
Tx #1	Tx #1
Tx #2	Tx #2
Tx #3	Tx #3
Tx #4	Tx #4
Tx #5	Tx #5
Tx #6	Tx #6
Tx #10	Tx #15

1. Alice estimates diff set size
2. Alice computes a sketch of her set
3. Alice sends the sketch to Bob
4. Bob computes his sketch
5. Bob XORs sketches
6. Bob can find Tx #10 and Tx #15

Minisketch: Computing a BCH sketch

$$Elements : a_1, a_2, a_3, a_4, a_5, \ldots, a_m$$

$$Summary(Syndromes) :$$

$$S_1 : a_1 + a_2 + a_3 + a_4 + a_5 + \ldots + a_m$$

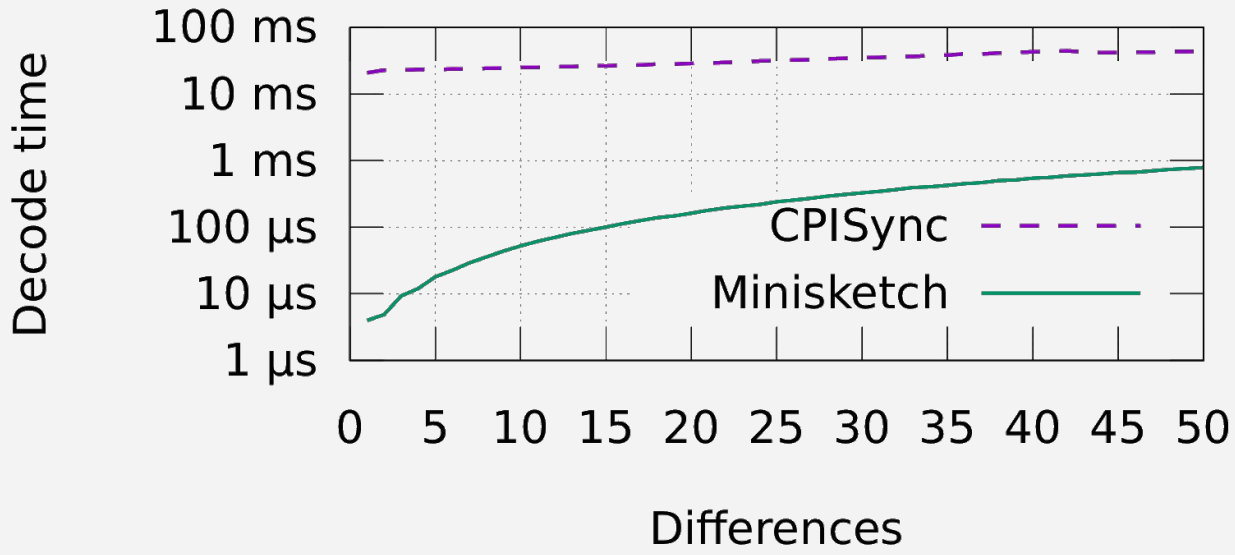
$$S_2 : a_1^2 + a_2^2 + a_3^2 + a_4^2 + a_5^2 + \ldots + a_m^2$$

$$\ldots$$

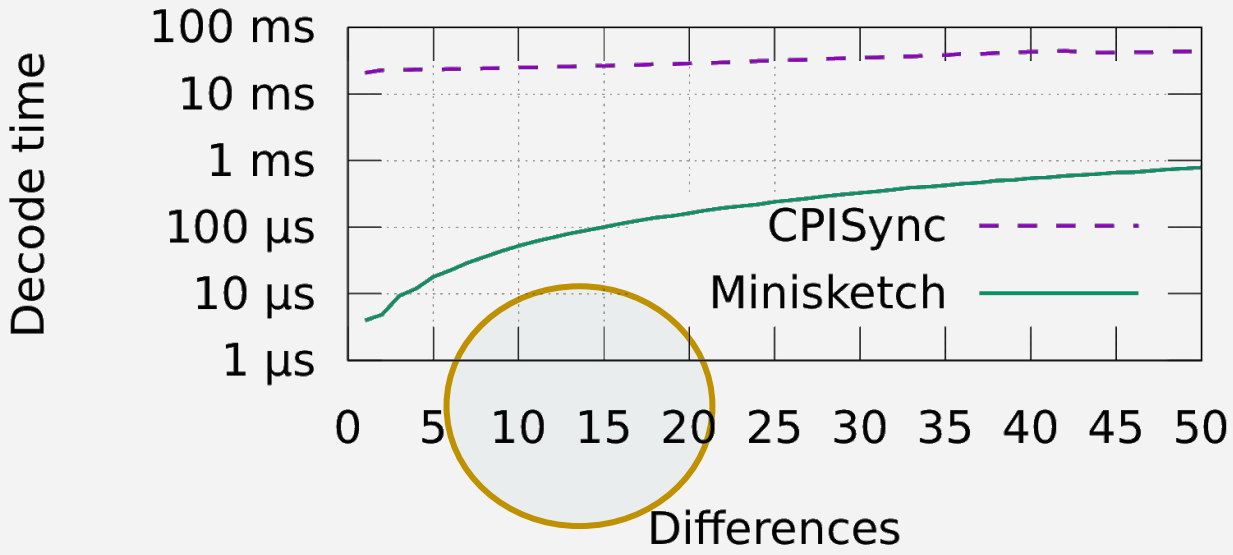
$$S_N : a_1^N + a_2^N + a_3^N + a_4^N + a_5^N + \ldots + a_m^N$$

* N — assumed max difference size

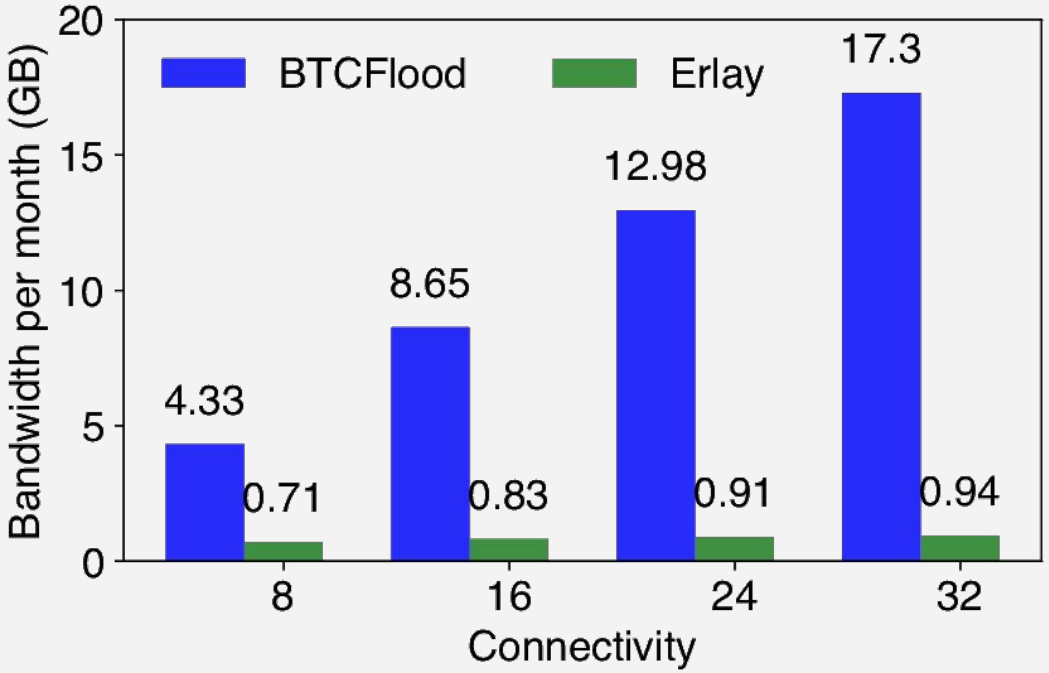
Minisketch (PinSketch implementation) benchmark



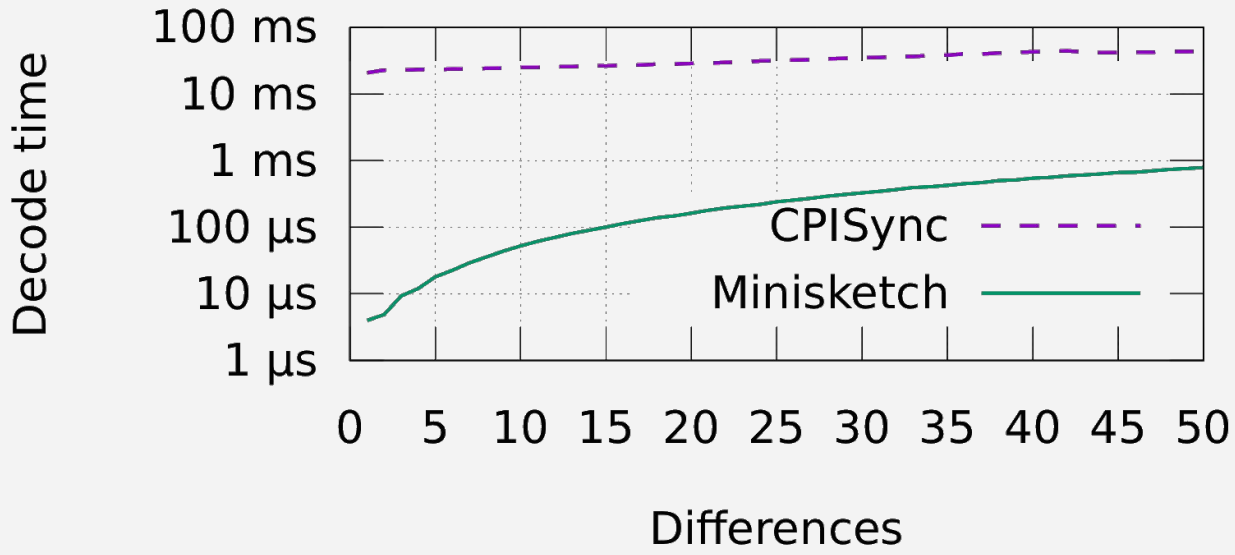
Minisketch (PinSketch implementation) benchmark



Erlay bandwidth benchmark. Simulation

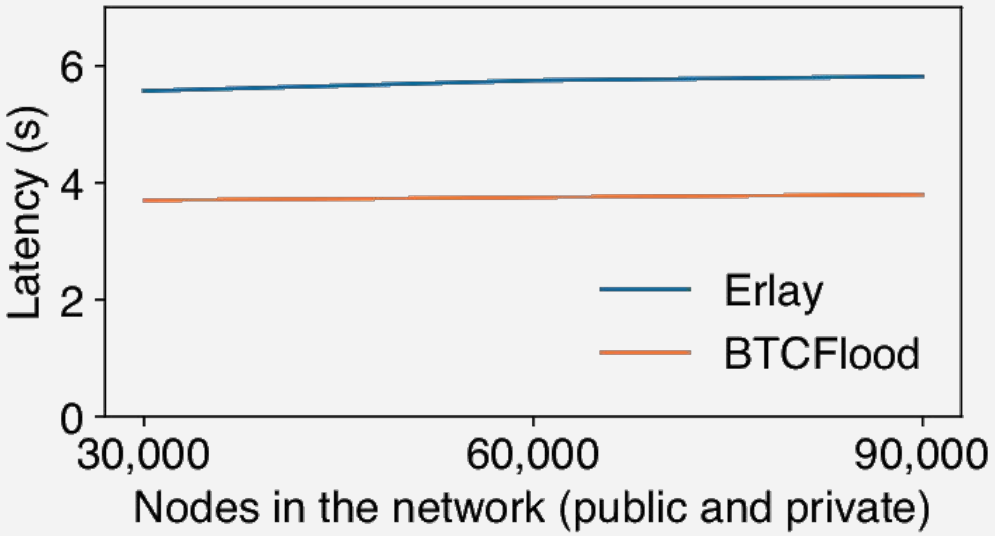


Minisketch (PinSketch implementation) benchmark



Erlay latency benchmark. Simulation

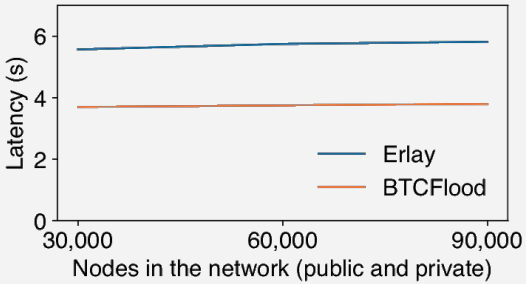
Time it takes for a protocol to relay across all nodes



Does latency increase matter?

Transaction relay latency increase might potentially

increase stale block rate, **bad for the security of the network**



chaincode

Does latency increase matter?

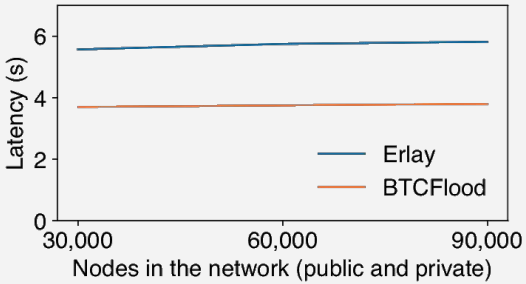
Transaction relay latency increase might potentially

increase stale block rate, bad for the security of the network.

Because of the faster relay across *public nodes*, stale block rate with Erlay

is actually **lower**!

Good for the security of the network.

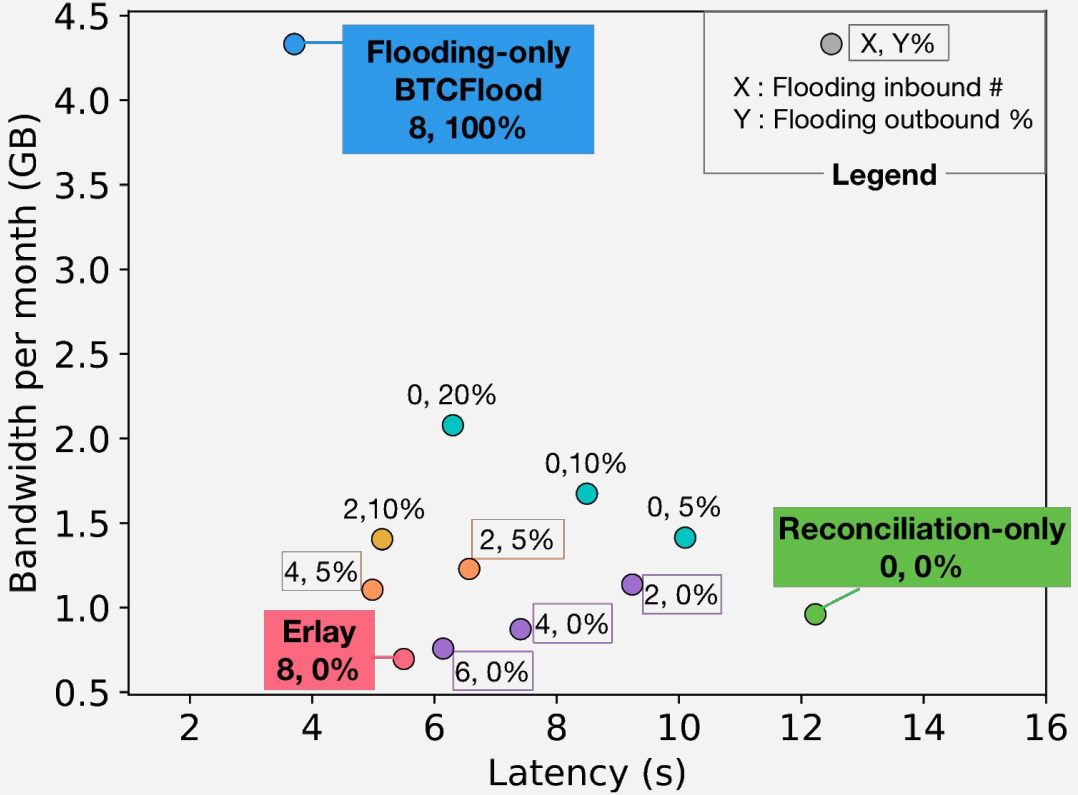


Erlay benchmark. Prototype

100 Azure machines running Bitcoin Core software and relaying 500 transactions

	BTCFlood	Erlay
Base cost (MB) (TX+GETDATA)	27	27
Other messages (MB)	1.06	1.1
Announcement cost (MB)	42	15
Latency (s)	1.85	2.05

Various configurations of the Erlay-like protocols



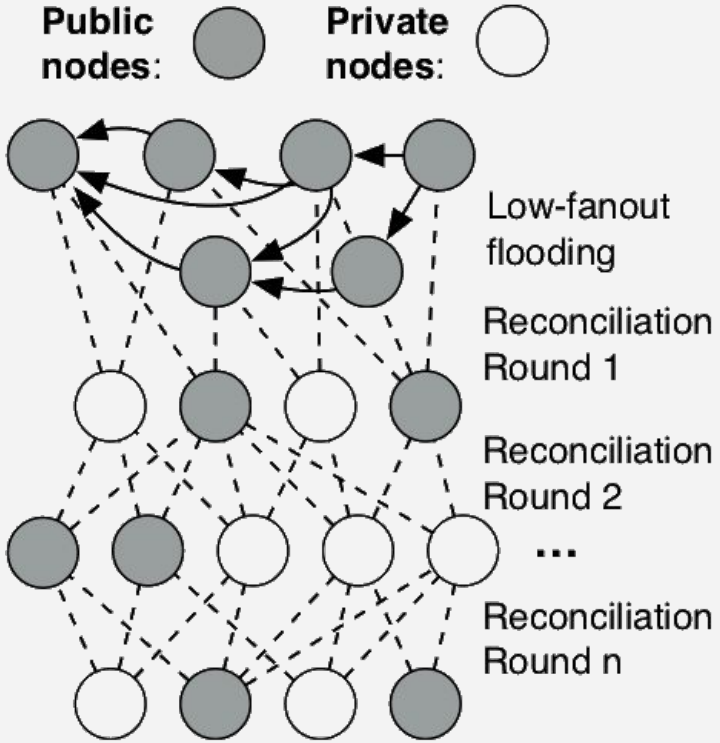
Erlay: Efficient transaction relay

Pros:

- Saves ~40% bandwidth for every node
- Allows to increase connectivity for free
- Stronger privacy (first-spy estimator)

Cons (not really):

- Minor latency increase



Erlay: Efficient transaction relay

Gleb Naumenko, Greg Maxwell, Pieter Wuille, Sasha Fedorova, Ivan Beschastnikh

Full paper:
https://arxiv.org/abs/1905.10518

Email: naumenko.gs@gmail.com
Twitter/Telegram @tomatodread
Bitcoin-IRC: gleb