I’m working on designs for a transaction gossip network which is conducive to participation by lightweight clients.

If you want to read about the idea in depth, here is my WIP document I’m using to capture my thoughts: https://notes.ethereum.org/YtI7e-R5RSK13cklq8j3Ug?both

Design Goals

- Interested parties can maintain a view of the full transaction pool (like miners or arbitrage bots)
- Nodes can tune

their level of participation down to the amount of resources they have available.

- Preservation of DOS mitigation mechanisms.
- Transaction validation can be done statelessly

Transaction Radius based Gossip

The network is based on long lived connections between nodes (such as DevP2P or LibP2P)

Each node on the network has a node_id

Each transaction in the pool has a transaction_hash

We define the distance

between a node and a transaction to be abs(node_id - transaction_hash)

with node_id

and transaction_hash

interpreted as big-endian integers. (an alternate distance metric like xor would be fine as well, absolute distance is just simple to reason about)

Each node on the network publishes a radius

. Connected nodes should only publish transaction hashes that fall within a peer’s published radius

.

Nodes actively prioritize peer connections using these rules:

- Mutual Interest: Maximize overlap between the segment of the transactions we are interested in and the segment they are interested in.
- Similar Resources: Prefer nodes with a radius

value that is close to ours.

Transactions are gossiped with an attached proof of the account balance and nonce.

Network Properties

These rules result in an emergent network topology with the following properties:

Tuning resource requirements with radius

Nodes with a radius of 1

are committing to managing the full pool. Users like miners or front runners.

Other full nodes might use a radius value of something like 0.2

in order to limit themselves to only 40% of the pool’s transactions.

Very lightweight nodes might use a radius value of 0.005

to only process 1% of the transactions.

Center vs Edge performance

Since nodes end up preferring connections with other nodes with a similar radius, the emergent topology is that there will tend to be more hops between nodes with a large radius and nodes with a small radius.

This means that transactions which originate from a node with a very small radius will take longer to reach the “miner” nodes who have very large radius values.

The other emergent property from this is that nodes with a small radius are “easy” peer connections to manage since you should only need to send them a small set of transactions. Similarly, nodes with larger radius values require more resources since they must be sent a larger number of transactions.

Combating “line jumping”

Some nodes may try to “lie” about their radius, publishing a large radius but not actually relaying transactions. The motivation would be to maintain a connection to nodes with a full radius without putting the necessary resources towards actually processing the transactions. This behavior should be easy to detect, since such a node would not be publishing transactions for gossip.

DOS Issues

This network design introduces two new mechanics that are relevant to DOS protections.

- When the network has two transactions from the same sender with the same nonce

but different hashes one of them needs to be discarded as spam. Lightweight nodes who’s radius does not cover both transactions will not have visibility into both transactions and thus cannot easily detect this situation.

- The attached proofs can become outdated and lightweight nodes cannot easily update them.

Both of these issues result in the “edges” of the network being more susceptible to DOS attacks. Intuition suggests that this should not effect the “core” of the network since the spam will be filtered out as soon as it reaches a node with enough visibility to see the other conflicting transactions. It should be ok for nodes at the edge of the network to drop transactions with outdated proofs.