Routing Graph Reconcilation 0.0.1

Piotr Mikulski

1 Overview

In P2P networks, like NEAR Protocol [1], each node stores a graph of nodes and edges representing peers and connections between them. We will call such graph a **routing graph**. Whenever a peer joins the network or peer reconnect, they exchange their routing tables. The process of exchanging tables involves adding edges, which are present in one's peers view of the graph, but not in the others. For the same of simplicity we will describe only the process of adding new edges.

That operation gets more expensive as **routing graph** grows larger. Exchanging full routing tables on every reconnect would be expensive and we would like to avoid it.

In this article, we will discuss ways of exchanging routing tables by echanging the amount of information proportional to size of edges that need to be added.

2 Definitions

2.1 Routing Graph

Graph, where peers are represented by nodes, and edges by connections between them.

2.2 Node

Each node is uniquely defined by it's public key.

```
pub struct PeerId {
    /// Peer is defined by it's public key
    public_key: PublicKey,
}
```

2.3 Edges

Each edge is defined by pair of peer ids and signatures from both peers. This data structure can be as large as 360 bytes in case of NEAR Protocol network. Therefore we would like to minimize the number of edges we transfer.

```
pub struct Edge {
    /// Since edges are not directed `peer0 < peer1` should hold.
    pub peer0: PeerId,
    pub peer1: PeedId,
    /// Signature from parties validating the edge.
    These are signature of the added edge.
    signature0: Signature,
    signature1: Signature,
    /// some other data
    /// ...
}
```

2.4 IBF - Inverse Bloom Filter

Invertible Bloom Filter (IBF) is a type of bloom filter, which can be used to simultaneously calculate D_{A-B} and D_{B-A} using $O(d)$ space. This data structure encodes sets in such a way that given two encodings of two different sets, you can recover symetric difference of those two sets. It can be shown that given two IBF structures of size k, you can recover with high probability up to $(2/3) * k$ elements if k big enough. Detailed analysis can be found at ESRwPC [4].

3 Proposed algorithm

4 Performance results

5 References

References

[1] NEAR Protocol *https://near.org/*

[2] Efficient Set Reconciliation without Prior Context. *https://www.ics.uci.edu/ eppstein/pubs/EppGooUye-SIGCOMM-11.pdf*

[3] Minisketch: a library for BCH-based set reconciliation *https://github.com/eupn/minisketch-rs*

[4] - [4] Efficient Set Reconciliation without Prior Context *https://www.ics.uci.edu/ eppstein/pubs/EppGooUye-SIGCOMM-11.pdf*