May 2, 2021

1 Bitcoin peer-to-peer protocol

Introduction to btc network, has to glue together all the parts of this section. Clients versions/ variants - how different clients interact - networking interface almost untouched. This thesis refers to latest version of bitcoin core.

1.1 Connection and peer discovery

In order to establish a connection two peers need to exchange version messages on a TCP channel, by default on port 8333, in an handshake fashion. The node that first sends a `version` message is said to be establishing an *outgoing connection*. The receiving peer will eventually send back a `verack` message followed by another `version` message. This latter node is said to be establishing an *ingoing connection*. Once the second `version` message is answered with a `verack` the connection is set up. Each node is able to establish up to 8 outgoing connections and 117 ingoing connections, for a total of 125 connections. In order to keep its connections active, each peer sends a message to each neighbour at least once every thirty minutes. If more ninety minutes pass without receiving anything from a neighbour, the connection is dropped. Each node always tries to keep eight outgoing connections active, therefore upon dropping an outgoing connection a node tries to connect to another address from its peer cache.
Nodes on fresh bootstrap make use of hardcoded *DNS seeds* as their first mean to discover other peers. DNS seed servers are maintained by community members and answer queries providing a list of addresses that can be either dynamically gathered through a periodic scan of the network or manually updated by server administrators. As a fallback option the user is able through command line to specify a list of addresses the client can connect to. Were both these two options to fail the client has a hardcoded list of peers it can directly connect to, though this is considered the last resort for bootstrapping peers. Nodes on startup that were previously in the network shall first lookup for peer names in their local address database, implemented as "peer.dat". The database contains the address of each peer the node has come to know during its lifetime in the network. If the node has disconnected for a time too long, many of the addresses in the database may have become outdated or unreachable. A node that cannot connect to any address in the peer database or has spent up to eleven seconds trying to connect unsuccessfully to at least one of the peers in the database behaves as on fresh bootstrap and resolves to query a DNS first. The use of a local address database, also called *peer cache*, provides reconnecting peers a fully-decentralized way to join the network and is the first line of defense against *fake bootstrap attacks*; the

topic along with other security concerns will be discussed in Chapter $< num >$.
The peer discovery process after the first connection of a node is carried on through the exchange of `addr` messages containing the address and port number of other peers in the network. Once a connection has been set up the two nodes exchange `addr` messages, providing each other with addresses from their local peer database. Furthermore every twenty-four hours each node sends to its neighbours an `addr` message containing only its own address in order to advertise itself in the network. `addr` messages can contain at most one thousand addresses; additionally those containing ten or fewer addresses are relayed. This behaviour greatly contributes to the gossiping of self-advertisement messages even though messages are relayed only to a small subset of neighbours, namely to a couple of peers.

2 Bitcoin Network Security Issues

-network attacks on bitcoin
-atks that operate on peer cache
Peer caches have been widely implemented so far as they are a fast and fully-decentralized way to rejoin the network after a disconnection. They allow P2P systems to overcome other major issues at bootstrap time related to presence of a single point of failure, i.e. bootstrap servers, and are the most scalable and efficient solution when compared to other decentralized mechanisms such as *address probing* [1]. Nevertheless, the use of peer caches introduces new security concerns for the network: the manipulation of the peer database performed by one or more cooperating malicious nodes can easily lead to the monopolization of a node, or group of nodes, connections', thus creating a network partition. This condition, often referred to as the *eclipsing* of node, is inherently dengerous for it leaves the victim open to *double-spend*, *selfish mining* and *51%* attacks. More can be found in the following sections.

2.0.1 Peer database structure

The local peer database or cache serves the purpose of storing the addresses a node has come to know from `addr` messages and DNS seeds. The database is constituted of the `new` and the `tried` tables. The latter contains addresses of peers with whom the node has successfully established a connection in the past. It consists of 64 buckets that can store up to 64 unique addresses. (Buckets are selected through a hash on the IP address) - questa non so se cancellarla. The `new` table contains the addresses the node has not connected to and is therefore larger: it has 256 buckets of 64 addresses each.

2.1 Eclipse attacks

questo articolo è importante [2]. During an eclipse attack the victim is flooded with a moltitude of incoming connections from an attacker and fed with bogus network information, i.e. peer addresses, that will fill up the peer cache. Upon restarting with high probability the victim will form all of its eight outgoing connections with malicious IP addresses, thus finding itself eclipsed. As described in Section 2.0.1 this greatly enhances the efficacy and

impact of other attacks. In 2015, Heilman et al. widely describe eclipse attacks on the Bitcoin network and suggest many countermeasures to harden the network a few of which were implemented [2]. As part of the adoption of these countermeasures, the number of buckets in `new` and `tried` has been increased, the addresses have become deterministically hashed to a single slot inside a bucket and the heavy bias towards addresses with fresher timestamps when choosing new connections has been removed.

2.2 Fake bootstrapping

ciaociao [1]

References

[1] Jochen Dinger and Oliver P. Waldhorst. Decentralized bootstrapping of p2p systems: A practical view. In Luigi Fratta, Henning Schulzrinne, Yutaka Takahashi, and Otto Spaniol, editors, *NETWORKING 2009*, pages 703–715, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] Sharon Goldberg Ethan Heilman. Alison Kendler, Aviv Zohar. Eclipse attacks on bitcoin's peer-to-peer network. Cryptology ePrint Archive, Report 2015/263, 2015. `https://eprint.iacr.org/2015/263`.