On the Estimation of the Number of Unreachable Peers in the Bitcoin P2P Network by Observation of Peer Announcements

Matthias Grundmann Hedwig Amberg Hannes Hartenstein

Institute of Information Security and Dependability (KASTEL)
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{matthias.grundmann,hannes.hartenstein}@kit.edu

Abstract

Bitcoin is based on a P2P network that is used to propagate transactions and blocks of the blockchain. While the P2P network design intends to hide the topology of the P2P network to impede adversarial actions against peers, information about the topology is required to understand the network from a scientific point of view, to build realistic models used for simulations, and to use these models to optimize protocols of the P2P networks. Thus, there is a natural tension between the 'desire' for unobservability on the one hand, and for observability on the other hand. On a middle ground, one would at least be interested on some statistical features of the Bitcoin network like the number of peers that participate in the propagation of transactions and blocks. This number is composed of the number of reachable peers that accept incoming connections and unreachable peers that do not accept incoming connections. Despite not accepting incoming connections, unreachable peers open several outgoing connections to other peers and participate in the propagation of transactions and blocks. While the number of reachable peers can be measured, it is inherently difficult to determine the number of unreachable peers, exactly because one cannot connect to them. Thus, the number of unreachable peers can only be estimated based on some indicators. In this paper, we first define our understanding of unreachable peers and then propose the PAL (Passive Announcement Listening) method which gives an estimate of the number of unreachable peers by observing ADDR messages that announce active IP addresses in the network. The PAL method allows for detecting unreachable peers that use flags to indicate that they provide services useful to the P2P network. In conjunction with previous methods, the PAL method can help to get a better estimate of the number of unreachable peers. We use the PAL method to analyze data from a long-term measurement of the Bitcoin P2P network that gives insights into the development of the number of unreachable peers over more than five years from 2015 to 2020. Results show that about 31,000 unreachable peers providing useful services were active per day at the end of the year 2020. An empirical validation indicates that the approach finds about 50 % of the unreachable peers that provide useful services.

1 Introduction

In Bitcoin [9], transactions and blocks are propagated by peers that are connected in a peer-to-peer (P2P) network. The protocol used for the propagation of transactions and blocks affects the whole system's performance security: The propagation delay of blocks is connected to the number of forks, i.e. temporal inconsistencies in the blockchain [2]. The protocol's bandwidth consumption limits the maximum number of connections [10] which limits the resilience against eclipse attacks [7]. The protocol can also leak information about the creator of a transaction [4] and partial information about the topology of the network

[11, 6, 3]. Analyzing the protocol with regard to these aspects and validating proposals for improvements can hardly be done in the live P2P network. Thus, researchers and developers use simulations of the P2P network for these tasks [12, 10]. Such simulations require a model of the topology of the P2P network. However, the real topology of the Bitcoin P2P network is unknown to the public[1] because the protocol is designed to hide information about the topology that could support adversarial actions against peers [4, 7]. Therefore, simulations model the topology based on certain characteristics that can be measured or estimated without having a method for the measurement built into the protocol itself. One of these characteristics and the object of this paper is the number of peers that participate in the propagation of transactions and blocks.

To form the P2P network, each peer creates by default 10 outgoing connections to other peers. A peer is a running instance of a Bitcoin software that is connected to at least one other instance of a Bitcoin software. Peers are encouraged to connect to multiple peers to reduce chances of being victim of an eclipse attack [7]. Not every peer accepts incoming connections, e.g., because a peer is behind a NAT or a firewall or a peer is configured to block incoming connections. Thus, the peers can be categorized into two groups: reachable peers that accept incoming connections and unreachable peers that do not accept incoming connections. Despite the name, unreachable peers play an active role in the P2P network. While unreachable peers do not accept incoming connections, they open several outgoing connections to other peers and participate in the propagation of transactions and blocks just as reachable peers do. Hence, for the propagation of transactions and blocks both, the reachable and unreachable peers are relevant. Franzoni and Daza [5] recently presented how the robustness and efficiency of the P2P network can be improved by giving unreachable peers a special role in the propagation of transactions.

There exist projects[2] continuously measuring the number of reachable peers. However, unreachable peers are mostly invisible because one cannot connect to them. Although we can *define* the set of unreachable peers, we cannot *retrieve* it using measurements: We can only *estimate* the number of unreachable peers. There are two ways to get such an estimate: The first way is to observe a fraction of unreachable peers and extrapolate the whole number of unreachable peers. This can be done by running a reachable peer that accepts incoming connections from unreachable peers. The second way is to observe effects that are caused by unreachable peers and infer their number from these observations. In this paper, we present an approach that uses the second way to estimate the number of unreachable peers. This approach, that we call the PAL (Passive Announcement Listening) method, relies on observing address announcements that are forwarded by reachable peers in the network. The approach is only able to count unreachable peers that offer services to the network such as storing the latest 288 blocks of the blockchain. Addresses of unreachable peers that do not announce such useful services will not be forwarded by other peers and thus not be counted. As there is no ground truth available, we validate our approach by verifying our assumptions and by comparing the results of our approach to an observation of a fraction of unreachable peers. We show that the approach detects most reachable peers and about 50 % of unreachable peers that provide useful services. Previous work has estimated the number of unreachable peers to be around 16,000 peers [11], 54,000 peers [10], 100,000 peers [1], and 155,000 peers [15]. The wide range of differences comes not only from different measuring times and methods but also from the fact that different definitions of the term *unreachable peer* are used.

[1]Approaches to gain (partial) information about the topology have been proposed [1, 8, 11, 6, 3]. However, they either do not work anymore or require a strong attacker to get the complete P2P network's topology.

[2]E.g., `https://bitnodes.io/` and `https://dsn.kastel.kit.edu/bitcoin/`

Thus, we start by defining the term *unreachable peers* and other relevant terms in the following section. An overview of related work will be given in Section 3. We describe the Bitcoin protocol and the behavior of the most common Bitcoin implementation in Section 4. Then, we present the PAL method in Section 5 and present the results of applying the method to data collected from the Bitcoin P2P network between 2015 and 2020. We validate the method in Section 6 and conclude in Section 7.

2 Definitions and Problem Statement

We refer to an implementation of the Bitcoin protocol as Bitcoin software. As stated above, we define a *peer* as a running instance of a Bitcoin software that is connected to at least one other running instance of a Bitcoin software. We expect most peers, however, to be connected to multiple peers in order to reduce chances of being a victim of an eclipse attack. A Bitcoin P2P network consists of peers that are directly or indirectly connected to each other. Because this definition allows multiple Bitcoin P2P networks, we refer to *the* Bitcoin P2P network as the Bitcoin P2P network that includes the peers that mine blocks with more computation power than the peers in every other Bitcoin P2P network. In the following, we consider only peers that are part of *the* Bitcoin P2P network.

Categorizing peers into reachable and unreachable peers is more difficult than it might seem at a first glance. A first approach would be to define a peer as unreachable if all other peers cannot initiate a connection to that peer. A peer might be unreachable because it runs behind a firewall that blocks incoming connections. However, a peer might accept incoming connections from one group of peers but refuse incoming connections from other peers. Imagine a private network that blocks incoming connections from the outside to peers inside the network but allows for incoming connections between peers that are inside the private network (see Fig. 1a). Because a peer in such a network would accept incoming connections from *some* peers, this peer would not be called unreachable following the above definition of unreachable. However, this peer would be unreachable for all peers outside the private network and, thus, should by intuition be categorized as unreachable. Therefore, a definition of unreachable peers may not be too strict. We consider such cases in the following definition that we use in this paper: A peer p is called *unreachable* if the majority of other peers cannot initiate a connection to p. This means that a peer is called reachable if most other peers can initiate a connection to that peer. Note that, if the majority of peers were inside a private network that blocks connections from the outside but allows for internal connections, this definition would categorize the peers in this private network as reachable (see Fig. 1b) because the majority of other peers is in the same private network. Indeed, the peers in this private network might have many incoming connections and, thus, look like reachable peers. However, as peers outside the private network cannot initiate a connection to them, the peers in the private network would seem unreachable to the rest of the world. We think that a binary classification into reachable and unreachable peers would not be suitable for such a scenario. For this work, however, we assume that the Bitcoin P2P network follows mostly the model as sketched in Fig. 1a, i.e. most peers in the Bitcoin P2P network accept incoming connections from either all other peers or none.

Our goal is to estimate the number of unreachable peers. Because peers join and leave the network (*churn*) this number changes continuously. The number of peers at a given point in time can be different from the number of peers that existed during a given time period (e.g., during one day). In the following, we will talk about estimating the number of unreachable peers during time periods. Using a model for churn, this number can be used to estimate how many unreachable peers existed at a given point in time.

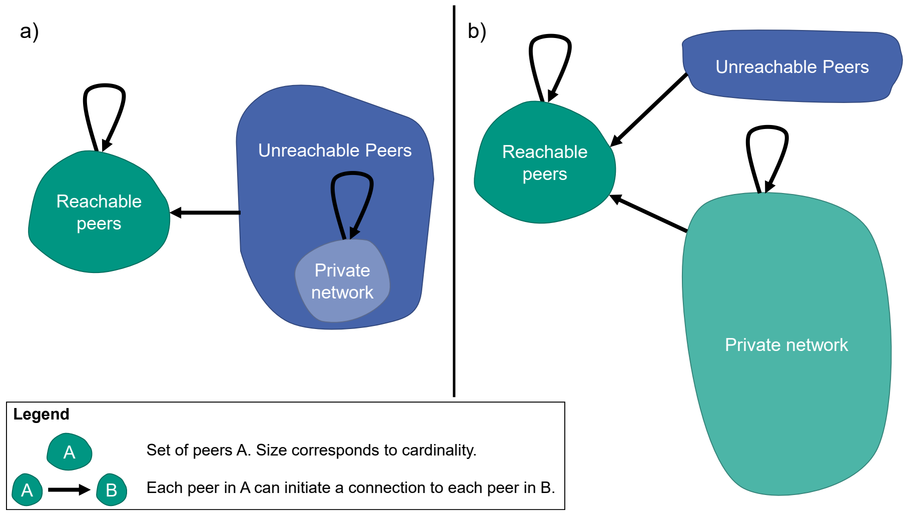

a)
Reachable peers
Unreachable Peers
Private network
b)
Unreachable Peers
Reachable peers
Private network
Legend
A
Set of peers A. Size corresponds to cardinality.
A
B
Each peer in A can initiate a connection to each peer in B.

Figure 1: Two simple models of how the peers in the Bitcoin P2P network might be connected. a) Each unreachable peer can initiate a connection to each reachable peer. A reachable peer can initiate a connection to another reachable peer. Additionally, there might be a private network of unreachable peers that can connect to each other. b) The number of peers in the private network might exceed the number of other peers. In this case, our definition would classify the peers in the private network as reachable peers although they are not reachable by peers outside the private network. This is not a problem for this work because we assume that the Bitcoin P2P network looks similar to the model a).

The peers in the Bitcoin P2P network are identified by their addresses. A peer can have multiple addresses (in the most common case an IPv4 address and an IPv6 address) and multiple peers can share an address (e.g., an IPv4 address because they are behind the same NAT). In this work, we will make the simplifying assumption that each peer has exactly one address. If we simply use the term address, then it refers to any type of address being used in the Bitcoin protocol, e.g., IPv4 address, IPv6 address, or Tor address (see BIP 155[3] for full list).

3 Related Work

The number of reachable peers is continuously measured by different projects (see Footnote 2 on page 2). They share the basic approach of recursively searching the network for peers. Exemplary, we explain the approach of Bitnodes[4] which is similar to that of [13]: The software starts with an initial set of peers, connects to each peer and requests addresses from this peer using a GETADDR message. On receiving an ADDR message as reply, the software tries to connect to each of the addresses in the reply and, for each successfully opened connection, addresses are requested over this new connection. The set of peers that a connection has been established to, is regarded as the set of reachable peers. In case a connection to an address cannot be established, it is unknown whether there is no peer at this address or there is an unreachable peer at this address. Therefore, this approach is not capable of measuring the number of unreachable peers.

In previous work, only few attempts have been made to estimate the number of unreachable peers. In May 2017, Wang and Pustogarov [15] ran 102 reachable peers as probes in

[3]`https://github.com/bitcoin/bips/blob/master/bip-0155.mediawiki`
[4]`https://github.com/ayeowch/bitnodes/`

Table 1: Messages of the Bitcoin protocol relevant for our study

Type	Relevant Fields
VERSION	protocol version, services, user agent, address of receiving peer, …
VERACK	–
GETADDR	–
ADDR	number of entries, for each entry: timestamp, services, address, port

different data centers around the globe for seven days and logged all incoming connections and associated information. For each peer that connected to one of the probes, they tried to open a TCP connection on port 8333 to that peer's IP address and to open a connection via the Bitcoin protocol. They observed on average about 10,000 unique IP addresses in a 6-hours interval. They assume an average of 3.5 connections per unreachable peer and 5,540 reachable peers and use these numbers to estimate that there were at least 155,000 unreachable peers in each 6-hours interval. The authors measured that 34.6 % of the observed connections lasted shorter than one second and 93.9 % lasted shorter than one minute.

To parameterize their simulation, Naumenko et al. [10] used numbers obtained from a website run by Luke-Jr[5]. They obtained the information that the network had 54,000 unreachable peers (called non-listening) and 6,000 reachable peers (called listening). At the time of writing (January 2021), the website lists about 50,000 unreachable peers and 5,000 reachable peers. The methodology behind the website is not publicly documented, but, in the absence of other reference points, we also compare our measurements to the numbers obtained from this website.

4 Bitcoin Peers

The protocol for peers in the Bitcoin P2P network is described by a public developer reference[6]. The protocol does not distinguish between reachable and unreachable peers because a peer cannot detect whether it is unreachable or is reachable but does not have any incoming connections, yet. Because most of the peers run the same software, Bitcoin Core (announced in VERSION messages as "Satoshi"),[7] the behavior of this software is the de facto specification of the protocol. In the following, we start by describing the protocol rules that all peers should follow and then describe the relevant behavior of Bitcoin Core in more detail.

4.1 Bitcoin Protocol

Table 1 gives an overview of the messages in the Bitcoin protocol relevant for this paper. We briefly explain these messages in this paragraph. Peers need to know the addresses of other peers to be able to connect to them. To this end, addresses are propagated in the Bitcoin P2P network using ADDR messages. An ADDR message consists of a header, the number of entries that follow, and one or multiple entries. Each entry consists of an address, a port, a timestamp, and service flags. The timestamp was originally meant to describe when that peer was seen last, however, the behavior of the reference software was modified to not

[5]https://luke.dashjr.org/programs/bitcoin/files/charts/historical.html

[6]https://developer.bitcoin.org/reference/index.html

[7]https://dsn.kastel.kit.edu/bitcoin/

leak which peers are currently connected [8]. The service flags describe the services offered and extensions implemented by the peer running at the address. The protocol allows ADDR messages to contain up to 1000 entries. ADDR messages can be sent unsolicited and they can be requested using a GETADDR message. A peer replies to a GETADDR message with an ADDR message.

When a connection between two peers is established, they send VERSION messages to each other that contain the peers' user agents and services. The receiving peer replies to a VERSION message by sending a VERACK message.

4.2 Bitcoin Reference Software

By default, Bitcoin Core opens eight outgoing connections for full-relay and two outgoing connections that are used only for relaying blocks but not for transactions and addresses. Thus, an unreachable peer maintains outgoing connections to ten other peers. A reachable peer can additionally have incoming connections. The default maximum number of all (outgoing and incoming) connections is 125. In case a new incoming connection fills the last available slot, Bitcoin Core will evict an existing connection based on different metrics such as duration of the connection, ping times, and amount of transmitted data.

Peers request addresses from other peers using the GETADDR message which is answered by an ADDR message. The reply contains at most 23 % of the addresses in the replying peers database and at most 1000 addresses. ADDR messages are also sent unsolicitedly: A peer announces its address to a connected peer once a connection has been established. Each peer also regularly announces its address to its connected peers (except those for block-relay only). The announcements are sent at random times following an exponential distribution with a mean of 24 hours. In contrast to replies to a GETADDR message, these announcements are forwarded so that peers in the network learn about other peers. To distinguish between forwarded announcements and an announcement of a peer's own address, we call the later a *self announcement*.

To tell whether an ADDR message was received unsolicitedly or in reply to a GETADDR message, Bitcoin Core checks the number of entries in an ADDR message. Because announcements of addresses are unsolicited, most of the messages with announcements are small messages with ten or less entries. Thus, Bitcoin Core considers the addresses in an ADDR message for propagation if the ADDR message contains ten or less entries. An address is only propagated if it meets the following requirements: (1) The service flags associated with the address need to have the NODE_WITNESS flag set and the NODE_NETWORK or NODE_NETWORK_LIMITED flags. (2) The timestamp associated with the address must not be older than ten minutes. (3) The address itself must be routable, i.e., it may not be from an IP address range that is reserved for private use. If the decision is for an address to be propagated, then it is sent to one or two connected peers.

5 PAL Method and Results

In this section, we present the PAL method and give details on the setup and data collection and the methodology for analyzing the data. We discuss the limitations of this approach and present the results of applying this method to data collected during a timespan of five years.

For the PAL method, we run a passive monitor node that is connected to all reachable peers. The monitor collects unsolicitedly sent ADDR messages that are received from its peers. The set of all addresses received throughout a day results in an estimate of the active

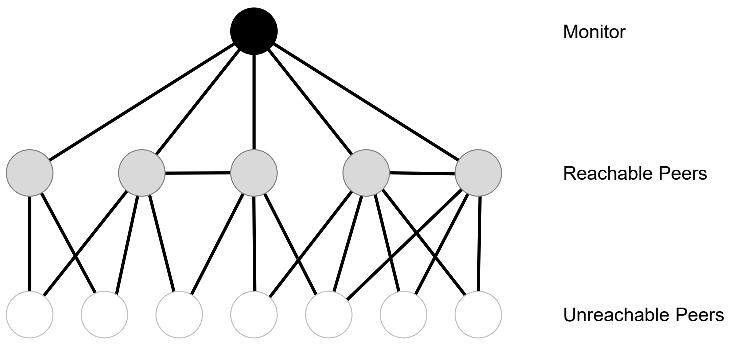


Figure 2: Overview of the setup. The monitor node is connected to all reachable peers. Unreachable peers are connected to reachable peers, too. There are connections between reachable peers but no connections between unreachable peers.

peers during this day. After removing the addresses of reachable peers from this set, we have an estimate of the set of unreachable peers.

The PAL method can be seen as a refinement on the approach for finding reachable peers explained at the beginning of Section 3. This approach itself is unsuitable for reliably finding unreachable peers because it only reveals that there are reachable peers at addresses at which an incoming connection is accepted. If no incoming connection is accepted at an address, the approach does not allow to draw any conclusions on whether there is no peer or an unreachable peer at this address. Instead of relying on GETADDR messages to quickly collect many addresses, we use a passive monitor node that waits for unsolicited ADDR messages. An address is only sent unsolicitedly if it is forwarded or if the sending peer announces this address as its own address (self announcement). If we receive an address in an unsolicited ADDR message at the monitor, we can conclude that at most ten minutes ago there was a peer at this address because these messages are only propagated until the timestamp associated with the address is older than ten minutes. Collecting all unsolicitedly sent addresses during one day gives us an estimate of the set of peers during this day. By filtering out reachable peers, we receive an estimate of the set of unreachable peers for this day.

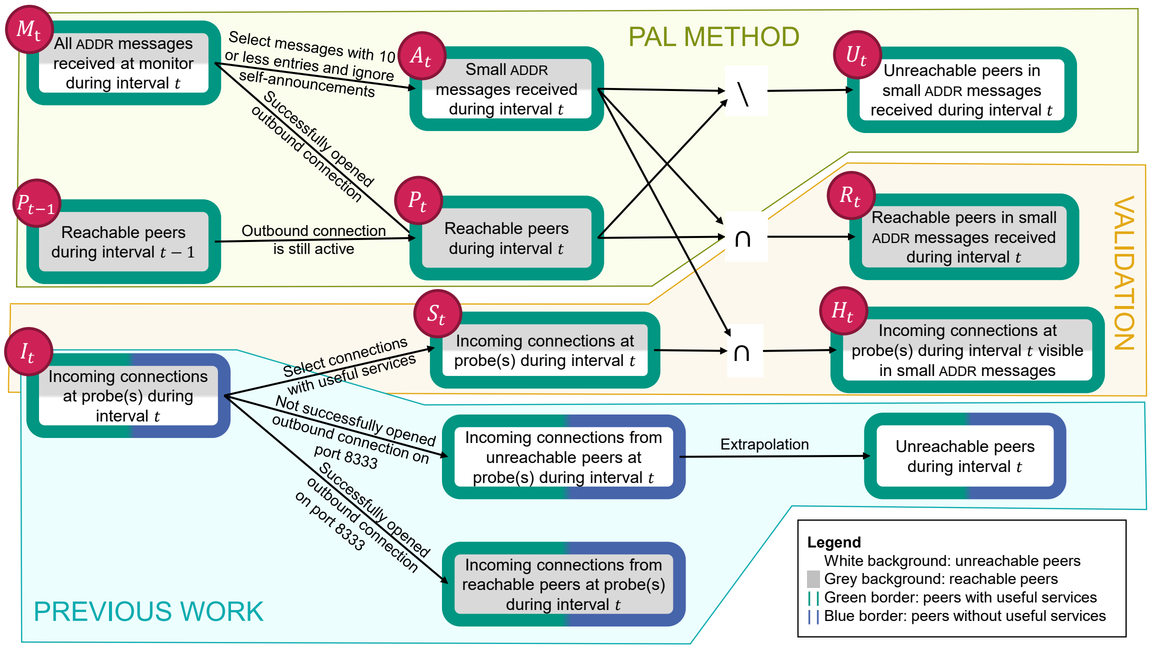
Data Collection We run a monitor node that connects to all known reachable peers in the network (see Fig. 2). The monitor is mostly passive, i.e. it does not send any messages with the following exceptions:

- After opening a connection, the monitor sends its own identifier in a VERSION message
- After having received another peer's VERSION message, the monitor not only sends a VERACK message but also requests addresses by sending a GETADDR message.
- Every two minutes, the monitor sends a GETADDR message to a uniform randomly chosen peer.

Sending the GETADDR messages is not required for the PAL method but also does not interfere with our results; the ADDR messages that are received in reply to these messages will be ignored in the following. The monitor tries to connect to each address it receives in ADDR messages (rate-limited per address to once every six hours). The monitor logs all received ADDR messages, VERSION messages and the time when a connection to another peer is established or closed.

Table 2: Overview of notation (also see Fig. 3)

Symbol	Description
A_t	Set of addresses received on day t in small ADDR messages (senders excluded)
P_t	Set of addresses that the monitor node was connected to on day t
U_t	(Estimation of) Set of addresses of unreachable peers on day t



PAL METHOD
VALIDATION
PREVIOUS WORK
M_t All ADDR messages received at monitor during interval t
Select messages with 10 or less entries and ignore self-announcements
A_t Small ADDR messages received during interval t
Successfully opened outbound connection
U_t Unreachable peers in small ADDR messages received during interval t
\
∩
P_{t-1} Reachable peers during interval $t-1$
Outbound connection is still active
P_t Reachable peers during interval t
R_t Reachable peers in small ADDR messages received during interval t
S_t Incoming connections at probe(s) during interval t
H_t Incoming connections at probe(s) during interval t visible in small ADDR messages
I_t Incoming connections at probe(s) during interval t
Select connections with useful services
Not successfully opened outbound connection on port 8333
Incoming connections from unreachable peers at probe(s) during interval t
Extrapolation
Unreachable peers during interval t
Successfully opened outbound connection on port 8333
Incoming connections from reachable peers at probe(s) during interval t
Legend
White background: unreachable peers
Grey background: reachable peers
Green border: peers with useful services
Blue border: peers without useful services

Figure 3: Data flow of the PAL method and the approach of [15] to estimate the number of reachable and unreachable peers in the Bitcoin P2P network. The sets M_t and I_t are collected during measurements and the arrows show filters and operations to derive more specific sets during the analysis. The border color of each box indicates whether the respective set contains peers with useful services only or also those without. The background color of each box indicates whether the respective set includes reachable and unreachable peers.

Data Analysis We analyze the logs created by the monitor with the goal of learning the number of peers in the network. We describe this process in the following and depict it in the upper part of Fig. 3. The most relevant notation symbols are listed in Table 2. For each day t, we collect all addresses that were received by the monitor (see M_t in Fig. 3). We define the set A_t by applying the following two filters on these addresses: (1) We select only the addresses in small ADDR messages (we define *small* for ADDR messages as containing ten or less entries). (2) We ignore the self announcements of (reachable) peers, i.e. entries of an ADDR message that equal the address of the sender of this ADDR message. The set A_t includes addresses of reachable and unreachable peers that were announced on this day. We determine the set P_t of all addresses that the monitor node was connected to on day t. For this, we collect all addresses that the monitor already was connected to at the beginning of day t or a connection was established and a VERSION message received during day t. We consider this set P_t as the set of all reachable peers at day t. Our estimate of the unreachable peers U_t for day t is $U_t = A_t \setminus P_t$.

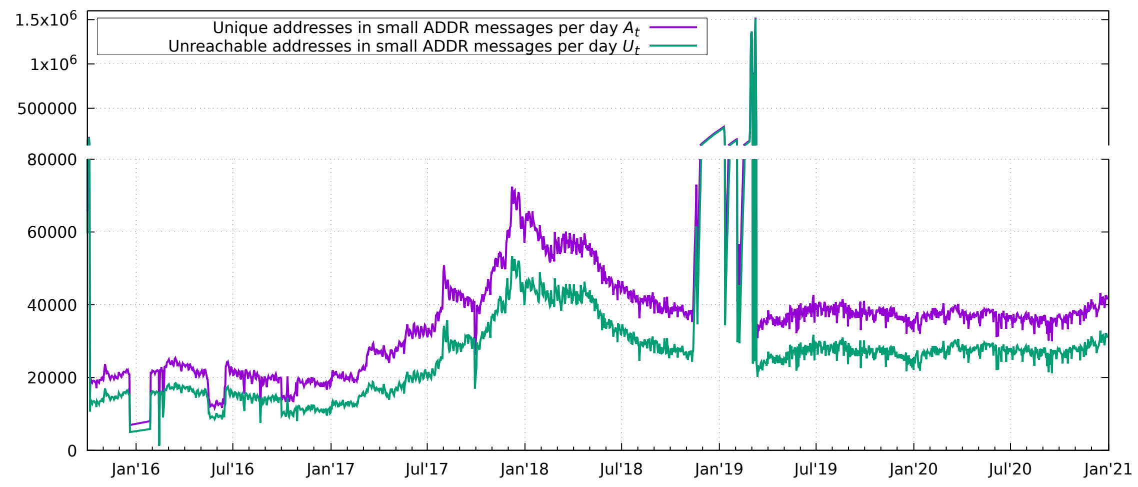


Figure 4: Number of unique addresses observed in small ADDR messages per day. Note that the upper part uses a different scale than the lower part.

Limitations As described in Section 4, peers running Bitcoin Core only forward addresses that have specific flags set (NODE_WITNESS and (NODE_NETWORK or NODE_NETWORK_LIMITED)). Thus, we do not expect to receive addresses of peers that do not have these flags set at the monitor node. The PAL method can only detect such peers with useful services which is indicated by the border colors in Fig. 3.

Another limitation of the PAL method is that it cannot distinguish whether an unreachable peer existed only for a short moment on a day or the whole day. Also, the addresses and associated information in ADDR messages are not authenticated. Thus, a peer can send any address and information about this address to other peers in the network. If a peer sends addresses in an ADDR message, there is no proof that the peer has received the address from another peer or is connected to a peer with this address. Therefore, the approach can be disturbed by flooding the network with bogus addresses.

Measurements We applied the method to data collected from 2015 to 2020 and present the results here. Figure 4 shows $|A_t|$, the number of unique addresses received in small messages for each day t and the number $|U_t|$ of addresses that were unreachable. The plot shows that, at the majority of days in the observation range, between 20,000 and 60,000 unique addresses were received in small messages. Most noticeably, the plot shows a high amount of addresses at the end of the year 2018 and beginning of the year 2019 which we will discuss later. The remaining plot shows that the number of addresses increased from the beginning of the year 2017 on to a maximum of about 72,000 addresses in the middle of December 2017. From this point on, the number of addresses declined slowly until the prominent peaks around the turn of the year 2018/2019 and since then the number of addresses has been relatively stable around 37,000. The number of unreachable peers $|U_t|$ has a similar development but is consistently about 28 % lower than the number of all addresses. At the end of the year 2020, the number of unreachable peers $|U_t|$ equals about 31,000 peers.

The peak at the end of the year 2018 seems like many unreachable peers joined the network within a few days. An alternative explanation would be that bogus addresses were distributed that do not actually belong to peers. We examined the addresses that were received only during this time and did not find any irregularities with regard to their distribution in the IP address space, autonomous system, or country of autonomous system.

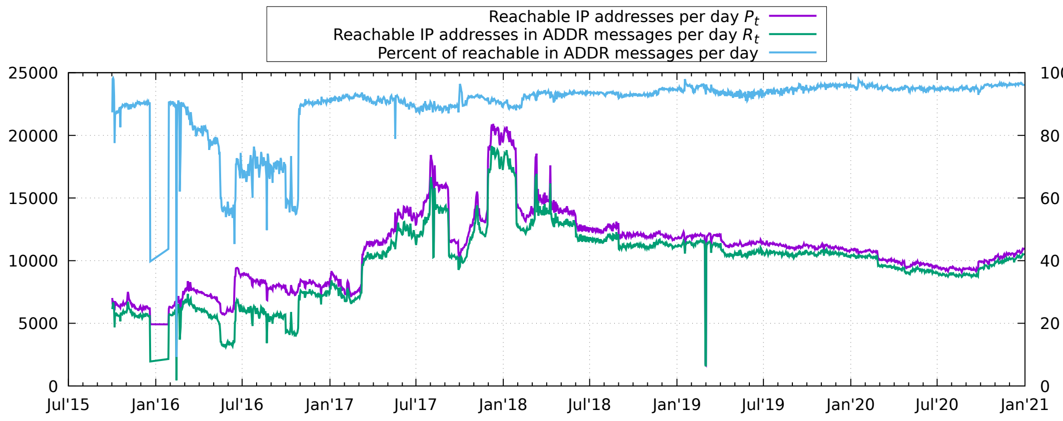


Figure 5: Number of unique addresses of reachable peers observed in small ADDR messages per day.

However, for the highest peak in March 2019, we found that this peak was caused by many IP addresses from the same /8 subnet. As IP addresses from this subnet were only very rarely observed before and after March 2019, we assume that this effect was caused by unknown actions of one party that flooded the network with these IP addresses. It is an open question whether a known or unknown attack would cause such effects and whether this can be verified using the data from our measurements.

6 Validation

To validate the PAL method, in this section we analyze the measured data and compare it to other sources.

6.1 Reachable Peers

As a first step, we verify that the results of the PAL method are consistent in itself. If our assumptions hold true, we should be able to find all reachable peers in small ADDR messages during each day. As we know the set of reachable peers, we can use it to evaluate the precision of our measurements. Putting this into the context of Fig. 3, this means that, if the PAL method works perfectly, we expect that set R_t equals set P_t.

In Fig. 5, we plot the number $|P_t|$ of reachable peers for each day t and the number $|A_t \cap P_t| = |R_t|$ of reachable peers whose addresses we received on each day t. The plot shows that since July 01, 2016 on each day on average 91.4 % of the addresses of reachable peers were received in a small ADDR message on the same day (excluding events for peers sending their own address). Over the time, the quality has increased: Since Jan 01, 2017 the average is 93.4 % and since Jan 01, 2020 the average is 95.3 %. This is a promising result as it indicates that reachable peers are consistently found by the PAL method. However, as can be seen in Fig. 3, we do not have an independent ground-truth for the set of reachable peers. Because all these sets of peers are based on the contents of ADDR messages, we can only conclude that the PAL method's constraint of looking at one day only and at small messages only does not reduce the set of detected reachable peers.

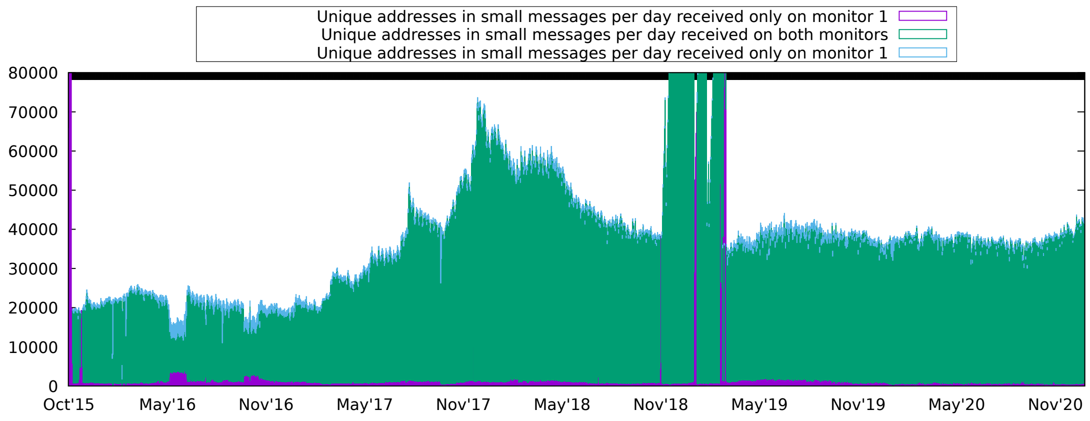


Figure 6: Stacked plot of the number of unique addresses observed in small ADDR messages per day at two monitors. The plot is separated into the number of addresses received only at monitor 1 (purple), the number of addresses received at both monitors (green), and the number of addresses received only at monitor 2 (blue). It can be seen that the majority of addresses is received at both monitors.

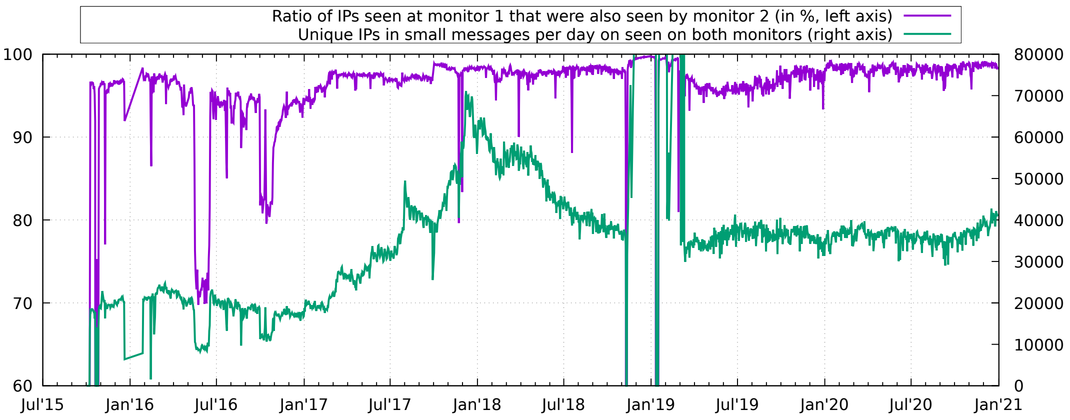


Figure 7: Ratio of IPs seen at monitor 1 that were also seen at monitor 2. The green plot shows the number of unique addresses observed in small ADDR messages per day at both monitors (right y-axis). The green plot is the same as the green plot in Fig. 6.

6.2 Second Monitor

For further validation, we run a second monitor node with the same method as describe above. We compare the addresses received by two monitor nodes. If the measurement method is reproducible, the data of both monitors should largely overlap. Figure 6 and Fig. 7 show the results for two monitor nodes for the whole time of measurement. It can be seen that since the beginning of 2017, over 95 % of the addresses observed at monitor 1 are also observed at monitor 2. This indicates that the measurement is reproducible and that the view of our monitor node is not subjective to the specific instance of the monitor.

6.3 Unreachable Peer

If our assumptions hold true, we can expect an unreachable peer that runs continuously to be visible in small ADDR messages every day. We run a small experiment to test this hypothesis. We start an unreachable peer p_U running the Bitcoin Core software in version v0.20.1 with the only modification to create a log entry when the peer announces its own address. The peer p_{U} has an IPv4 and an IPv6 address. The log entry consists of a timestamp, which address is announced, and which peer the address is announced to. The peer p_{U} runs continuously and it is unreachable because its incoming TCP port 8333 is blocked by a firewall. After being started, p_{U} creates ten outgoing connections (eight for full-relay and two for block-relay only). After the peer has been continuously running for 44 days, we analyze the logs and compare them to our monitoring logs. We first look at the number of announcements that the peer p_{U} has sent on each day. While one might expect that p_{U} sends on average eight announcements per day because it announces its address to every connected full-relay peer on average every 24 hours, the average measured is about 16. This is because although the peer p_{U} is running continuously, connections are closed by other peers and p_{U} creates new outgoing connections on which p_{U} announces its address. The number of announcements sent on a day is composed of the announcements on newly created connections and regular announcements on existing connections. This fact increases our expectations to receive an address of p_{U} at the monitor on every day. Next, we examine this proposition by looking for p_{U}'s addresses in the monitor's logs. We find that the monitor received p_{U}'s address on 43 of the 44 days. Assuming that this observation can be generalized, this indicates that the monitor sees the addresses of continuously running peers running the Bitcoin Core software with high probability on each day and the PAL method can detect these peers. We leave validation of this hypothesis for other Bitcoin software for future work.

6.4 Validation with Incoming Connections at a Public Peer

The approach of [15] is to run many public peers that accept incoming connections and observe the connections they receive which are partially from unreachable peers. For further validation of the PAL method, we use a similar approach and run a single public peer p_{I} that accepts incoming connections. We compare the collected data to the data found in ADDR messages to find out how many unreachable peers seen by p_{I} are also found using the PAL method.

The peer p_{I} does not open outgoing connections[8] and logs for each connection when it is established and closed and the received VERSION messages. We analyze the logs after p_{I} has been running for about 16 months. On average, the peer p_{I} had incoming connections from about 2,040 different addresses per day (this corresponds to the size of I_t in Fig. 3). As ADDR messages contain almost only addresses of peers providing useful services, we only select such addresses for comparison. The resulting set per day contains on average about 946 addresses (this corresponds to $|S_t|$ in Fig. 3). The intersection of this set with all addresses received in small ADDR messages on the same day, results in the set of addresses that opened a connection to p_{I}, announced useful services, and were observed in a small ADDR on the same day (see H_t in Fig. 3). If the PAL method worked perfectly, than this set H_t would equal the set S_t in Fig. 3. We find that on average 68 % of the addresses of incoming connections with useful services were also observed in small ADDR messages on the same day. As can be seen in Fig. 3, the set H_t contains reachable and unreachable peers. To validate the detection of unreachable peers only, we reduce this set to unreachable peers only by subtracting the set of reachable peers P_t. This shows that about 51 % of incoming connections of unreachable peers are detected by the PAL method in small ADDR messages. Analogously, we find that the PAL method detects about 98 % of incoming connections of reachable peers.

[8]Before the experiment, a peer existed that had the same address as p_{I} and initiated outgoing connections. Therefore, p_{I}'s address is contained in other peer's databases and peers open incoming connections to p_{I}.

These results show that the PAL method is very reliable to detect reachable peers and it detects about half of the unreachable peers with useful services. It is an open question why unreachable peers are not detected as reliably as reachable peers. Possible explanations could be that unreachable peers exist in the network for a shorter time than reachable peers, that unreachable peers use different software with a different behavior, or that they propagate an address that is different from the address they use for connections.

6.5 Comparison to Previous Measurements

As there is no ground truth that we could compare the data from the PAL method to, we compare it to measurements created by previous works. In 2014, Biryukov et al. [1] estimated that the Bitcoin P2P network had a size of 100,000 peers of which 90 % were estimated to be unreachable. This estimation is too early to be compared to our data but, taking the unreachable peers without useful services into account, this estimation could fit in with our observations. Neudecker et al. [11] simulated the Bitcoin P2P network in 2016 and estimated from the simulated propagation behavior that the P2P network had about 16,000 unreachable peers that participate in the propagation of transactions and blocks. The PAL method calculates about 14,000 unreachable peers per day averaged over the year 2016. As the results of [11] are given for one point in time and the PAL method estimates the number of unreachable peers during one day, we would rather expect that the PAL method would find more unreachable peers than [11]. Our explanation is that the lower number of unreachable peers detected might again be caused by peers not announcing useful services or by the effects mentioned at the end of Section 6.4.

A measurement of unreachable peers was conducted by Wang and Pustogarov in 2017 [15]. They ran more than 100 reachable peers to get incoming connections. From their measurements, they estimated at least 155,000 unreachable peers to be active in each 6-hours time period. This estimate is higher than the results obtained through the PAL method. An explanation for this is that Wang and Pustogarov report that 80 % of unreachable peers were mobile peers that had only short-lived connections. We assume that these peers either did not announce their IP addresses or that they did not provide useful services. In this case, they would be invisible to the PAL method which explains the difference to our results.

The measurement by Luke-Jr[9] gives an estimate of the number of reachable and unreachable peers over a similar timespan. In Fig. 8 we plot our data and the data from Luke-Jr in one plot for comparison. It can be seen that the number of unreachable peers in the data from Luke-Jr is higher compared to the number of addresses in small messages which is probably accounted for by the restriction that only addresses of peers providing useful services are propagated. The increase in addresses in small messages at the end of the year 2018 cannot be seen in the data from Luke-Jr which indicates that these peaks are not caused by many peers joining the network during this time. However, it seems that this period has had an effect on the measurements by Luke-Jr, too, because once the number of addresses in small messages had declined in March 2019, the number of unreachable peers as measured by Luke-Jr starts to rise. One interpretation of this fact is that the high number of ADDR messages at this time, hid unreachable peers from Luke-Jr's methodology and they became visible only after the number of ADDR messages had stabilized.

[9]`https://luke.dashjr.org/programs/bitcoin/files/charts/historical.html`

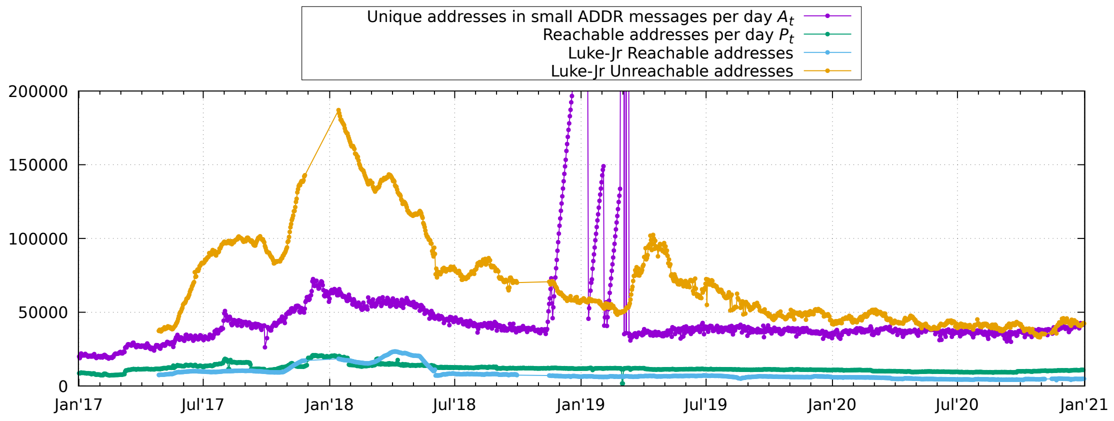


Figure 8: Comparison between the number of addresses observed in small ADDR messages per day and the data obtained from the website run by Luke-Jr.

7 Conclusion

We have presented the PAL method to estimate the number of peers in the Bitcoin P2P network including unreachable peers. The PAL method adds a further reference point to existing approaches measuring the number of unreachable peers. The estimate of the number of unreachable peers at the end of 2020 is about 31,000 peers which, as indicated by our validation, might correspond to about 50 % of all peers providing useful services. Looking at the measurements of the past five years, we have seen that the number of peers varied. As previous work considered only shorter time intervals, their measurements might have been during a time when the number of peers was higher than today. The insights gained from this work can be valuable for development and parameterization of simulators that model the Bitcoin P2P network more realistically.

To further improve on the PAL method and the validation, we plan to simulate the propagation of ADDR messages. We expect that this can enhance our understanding of the method and of effects that are visible in the measurements.

References

[1] Biryukov, A., Khovratovich, D., Pustogarov, I.: Deanonymisation of clients in Bitcoin P2P network. arXiv:1405.7418 [cs] (May 2014), `http://arxiv.org/abs/1405.7418`, arXiv: 1405.7418

[2] Decker, C., Wattenhofer, R.: Information propagation in the Bitcoin network. In: IEEE P2P 2013 Proceedings. pp. 1–10 (Sep 2013). https://doi.org/10.1109/P2P.2013.6688704, iSSN: 2161-3567

[3] Delgado-Segura, S., Bakshi, S., Pérez-Solà, C., Litton, J., Pachulski, A., Miller, A., Bhattacharjee, B.: TxProbe: Discovering Bitcoin's Network Topology Using Orphan Transactions. In: Goldberg, I., Moore, T. (eds.) Financial Cryptography and Data Security. pp. 550–566. Lecture Notes in Computer Science, Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-32101-7_32

[4] Fanti, G., Viswanath, P.: Deanonymization in the Bitcoin P2P network. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 1364–1373. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (Dec 2017)

[5] Franzoni, F., Daza, V.: Improving Bitcoin Transaction Propagation by Leveraging Unreachable Nodes. arXiv:2010.15070 [cs] (Oct 2020), http://arxiv.org/abs/2010.15070, arXiv: 2010.15070

[6] Grundmann, M., Neudecker, T., Hartenstein, H.: Exploiting Transaction Accumulation and Double Spends for Topology Inference in Bitcoin. In: Zohar, A., Eyal, I., Teague, V., Clark, J., Bracciali, A., Pintore, F., Sala, M. (eds.) Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 10958, pp. 113–126. Springer Berlin Heidelberg (2019)

[7] Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse Attacks on Bitcoin's Peer-to-Peer Network. Tech. Rep. 263 (2015), https://eprint.iacr.org/2015/263

[8] Miller, A., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N., Bhattacharjee, B.: Discovering bitcoin's public topology and influential nodes (2015)

[9] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System. Tech. rep. (2008)

[10] Naumenko, G., Maxwell, G., Wuille, P., Fedorova, A., Beschastnikh, I.: Erlay: Efficient Transaction Relay for Bitcoin. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security - CCS '19. pp. 817–831. ACM Press, London, United Kingdom (2019). https://doi.org/10.1145/3319535.3354237, http://dl.acm.org/citation.cfm?doid=3319535.3354237

[11] Neudecker, T., Andelfinger, P., Hartenstein, H.: Timing Analysis for Inferring the Topology of the Bitcoin Peer-to-Peer Network. In: 2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). pp. 358–367 (Jul 2016). https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0070

[12] Neudecker, T.: Security and Anonymity Aspects of the Network Layer of Permissionless Blockchains. Ph.D. thesis (2019). https://doi.org/10.5445/IR/1000089033, https://publikationen.bibliothek.kit.edu/1000089033

[13] Park, S., Im, S., Seol, Y., Paek, J.: Nodes in the Bitcoin Network: Comparative Measurement Study and Survey. IEEE Access **7**, 57009–57022 (2019). https://doi.org/10.1109/ACCESS.2019.2914098, conference Name: IEEE Access

[14] Tange, O.: GNU Parallel 20200522 ('Kraftwerk') (May 2020), https://doi.org/10.5281/zenodo.3841377

[15] Wang, L., Pustogarov, I.: Towards Better Understanding of Bitcoin Unreachable Peers. arXiv:1709.06837 [cs] (Sep 2017), http://arxiv.org/abs/1709.06837, arXiv: 1709.06837