Gotta Catch'em All! Improving P2P Network Crawling Strategies

Alexander Mühle(✉), Andreas Grüner, and Christoph Meinel

Hasso Plattner Institute, Potsdam, Germany
{alexander.muehle,andreas.gruener,christoph.meinel}@hpi.de

Abstract. Network crawling has been utilised to analyse peer-to-peer systems by academics and industry alike. However, accurately capturing snapshots is highly dependant on the crawlers' speed as the network can be described as a moving target. In this paper, we present improvements based on the example of a newly developed Bitcoin crawler that can be utilised to reduce resource usage/requirements of crawlers and therefore speed up capturing network snapshots. To evaluate the new strategies, we compare our solution, in terms of increased scan-rate and increased hit-rate during crawling, to a popular open-source Bitcoin monitor. Blocking time is reduced on average to 1.52 s, resulting in 94.7% higher scan-rates, while time needed to capture a network snapshot is reduced on average by 9% due to increased hit-rates during network crawling. While we show our improvements at the example of a new Bitcoin crawler, proven concepts can be transferred to other P2P networks as well.

Keywords: Peer-to-peer systems · Blockchain · Network crawling · Internet measurement

1 Introduction

Peer-to-Peer applications have seen a resurgence of popularity in recent years, often in the context of cryptocurrencies such as Bitcoin. Due to this trend, the research community, as well as commercial interests, have undertaken numerous projects to measure these kinds of networks.

Crawling networks is an inherently progressive process, yet the goal is to capture a snapshot of the current network and its participants. Ideally, a snapshot would include all current online nodes exactly. This process, however, is impacted by several factors.

We will describe but also quantify the most prominent of these factors such as churn, NATs, random address responses and lingering offline nodes in Sect. 2.

As main contribution in this paper we further the quest to an accurate network snapshot of large open peer-to-peer systems by improving the strategies used for such crawling. These include:

P. Gladyshev et al. (Eds.): ICDF2C 2021, LNICST 441, pp. 313–327, 2022.
https://doi.org/10.1007/978-3-031-06365-7_19

- Scale-out through a manager/worker architecture
- Optimal timeouts through network measurements of a live network
- Reduction of predictably unsuccessful connections by:
  - Taking into account previous offline time
  - Bogon filtering

We developed a Bitcoin crawling tool incorporating these improvements. We chose the practical example of the Bitcoin network as a current and popular peer-to-peer network and implemented a publicly available monitor of network participants. We utilised the gathered data for other published research on the Bitcoin network, such as characterising proxy usage in the Bitcoin peer-to-peer network [15].

We evaluated the effects of the improvements to the crawling in terms of increased scan rate, the rate at which probes are sent by the crawler, and increased hit rate, the fraction of positive responses by probed peers. Our improvements result in a 94.7% higher scan rate and 9% overall faster snapshot times using the same resources.

2 Motivation

Peer-to-peer network crawlers and recently Bitcoin crawlers are not new and are often utilised for research on the properties of the participants, topology and system behaviours overall. We recognise, however, that improving the speed and resource usage of crawlers is inherent to increased accuracy of such crawlers. This relation between accuracy of crawled snapshots and the speed/resource usage of crawlers is due to properties of peer-to-peer networks and especially privacy aware networks such as Bitcoin. In the following we will discuss these properties and quantify their magnitude in our chosen example network of Bitcoin with a preliminary study.

2.1 Churn

Typically crawlers join a given network and participate in the node discovery protocol, recursively contacting all other discovered peers. Churn, the effect of peers constantly leaving and joining such open networks, is a key contributor to inaccuracies of these measurements. The network can be described as a moving target, during the duration of the crawl, it is already changing, and when finished will not exactly represent the real state of the network. Due to this continuous fluctuation, the speed of a crawl has a direct and significant impact on the accuracy.

There are a few considerations we have to take into account in order to quantify the churn of the Bitcoin system accurately. The measurement of churn itself is impacted by the speed of the crawl and measurement. As we are not doing continuous probing but rather round based measures for our churn analysis, the granularity of the measurements is important. Missing data in between the crawls is an inaccuracy, however we believe that our granularity of roughly 3 min is enough

to have an understanding of the magnitude of churn in the network. Not only the granularity of the data but also the duration of the measurement is important as Stutzbach et al. noted in the their seminal paper on understanding churn in peer-to-peer networks [21]. Stutzbach et al. noted that while we can compute the duration of sessions that start and end during our measurement period, this would bias our statistics towards shorter sessions. We therefore chose our measurement duration as 5 days (also eliminating the impact of an diurnal effect), however for our churn statistic we will only take into account sessions that have started in the first 2.5 days. This is the so called "create-based method" used by Saroui et al. [19] in their analysis of the P2P networks Napster and Gnutella. In Fig. 1 a histogram of the observed session lengths during our initial study is shown. The y axis is of logarithmic scale as the session length follows a long tail distribution. Only a small fraction (0.8%) of discovered nodes were online for the complete observation time of 5 days. This is comparable to other previous studies of churn in the Bitcoin network that found the always online nodes to be 2.4% in 2018 at the height of Bitcoin popularity [9]. The median of sessions lengths has stayed the same even through the hype cycle of Bitcoin at one hour.

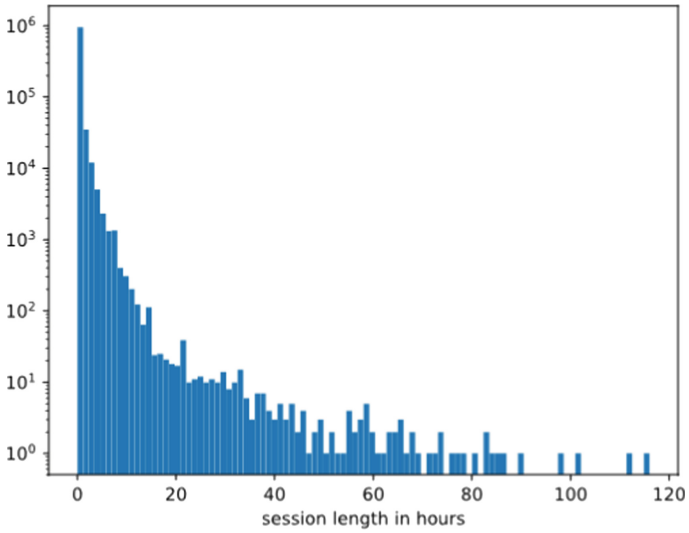

Fig. 1. Histogram of observed session length

2.2 Random Address Responses

For some networks, especially in privacy aware cryptocurrency networks such as Bitcoin, crawling participants in the network is not straight forward in order to minimise the attack surface. The responses to address requests are random and only a portion of the known peers of a contacted peer are returned to avoid revealing information to "network spies" (topology inference, eclipse attacks,

...). This also means repeated crawling is required to increase the likelihood of a more complete view of the network. The reference implementation of Bitcoin[1] randomly chooses 2500 addresses to return for address requests. As the maximum number of peers in the peer database of a client is 20480 we calculate the probability of not receiving a certain address in x requests as $1-(1/\frac{20480}{2500})^x$. The relation between number of requests and probability of not receiving a certain address is shown in Fig. 2.

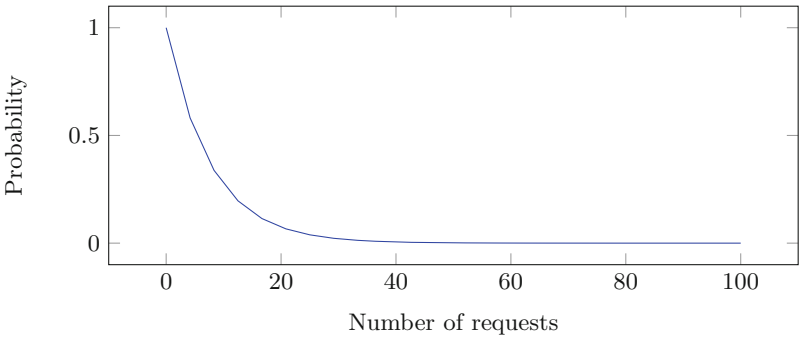

Fig. 2. Probability of an address not being discovered in relation to the number of requests

After a moderate amount of requests the probability quickly trends towards zero, Biryukov et al. estimate that 80 address requests are needed to reliably learn all addresses [1].

2.3 Unreachable Nodes

Generally the majority of peers will be unreachable by crawlers, either due to exiting the system or being unreachable in general. Nonetheless they will be discoverable from the peer databases of other participants. As a consequence, during crawling rounds, these addresses waste resources of potential crawlers through connection attempts that will timeout and are therefore costly. Network address translation has been recognised as a major contributor for unreachable peers in peer-to-peer settings. In 2009 Acunto et al. measured the NAT and firewall characteristics in peer-to-peer systems [6] and concluded that 90% of peers are behind a NAT or firewall with the trend of NATted peers increasing in future years. We confirm this by calculating the fraction of never-online peers in our preliminary Bitcoin study at 94%. Another contributing factor to unreachable nodes is the previously discussed churn. In most peer-to-peer networks, there are no explicit exits from the network, peers simply disconnect from their respective neighbours; this change, however, is not widely propagated. This leads to addresses of such nodes often lingering in the network for a considerable time.

[1] github.com/bitcoin.

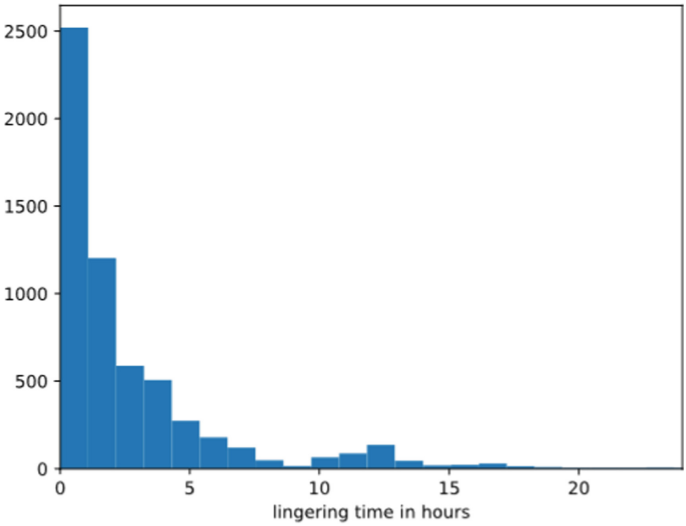

Fig. 3. Histogram of observed lingering time of offline nodes

Figure 3 shows the observed lingering time in the Bitcoin network. It is calculated by observing how long addresses are still included in address request responses after we have found them to be offline. The median lingering time is 1.43 h while the average is considerably longer at 3.7 h while some outliers were lingering for multiple days and as long as our preliminary study ran.

3 Related Work

Since the first large wave of peer-to-peer applications, the exploration of the associated network has been of interest to researchers. In the following, we will hence discuss both efforts from the general peer-to-peer context, often with the use-case of file-sharing, and more recent efforts looking at cryptocurrency networks. However there has also been an interest in network scanners from a more general purpose perspective which some aspects can be applied to peer-to-peer and cryptocurrency crawling.

3.1 Internet-Wide Scanning

Especially for security applications, internet-wide scanning and surveying of open ports has been utilised. The two most prominent examples of such tools are ZMap [5] and NMap [12]. While NMap is a more general purpose tool, ZMap focuses on single-packet probes which drastically increases the scan rate and design decisions such as the lack of state per-connection reduces the required resources for a large scale network scan. An interesting difference between NMap and ZMap design is the choice of no retransmission by ZMap, Durumic et al.

estimate that ZMap nonetheless achieves 98% network coverage using single packet probes per host. A key difference compared to the kind of peer-to-peer crawler we develop is that ZMap does not need to interact with hosts on the application layer, as hit lists (the list of hosts to be probed) are pre-configured and don't dynamically have to be generated through address discovery in a peer-to-peer network.

3.2 Peer-to-Peer Crawler

Gnutella was one of the first widely analysed peer-to-peer networks. Ripeanu et al. underwent a study to map the Gnutella network [18]. The paper tackles two main questions in regards to the then relatively new approach of peer-to-peer networks. Fault tolerance and robustness of the network as well as exploring the mismatch between virtual overlay network and physical internet infrastructure. For this purpose, they developed a crawler that uses the membership protocol of Gnutella to collect peer information. They utilised an initial list of nodes to contact, which then progressively gets extended by neighbour information of newly contacted peers. In order to speed up the crawling process, Ripeanu et al. used a manager/worker architecture where the server was responsible for assigning IPs to clients to contact.

In a very similar fashion, Stutzbach et al. [20] developed *cruiser* which they aimed to be an improved network crawler in order to capture more accurate snapshots of the Gnutella Network. They saw five key areas where this performance increase could be achieved. Inherent to the Gnutella protocol were two areas: handshaking and the two-tier structure of the Gnutella network. Just as Ripeanu before them, they also deployed *cruiser* as a distributed system with manager/worker architecture. Finally, they recognised that appropriate timeouts were essential for a performant crawler as non-responsive peers were a significant percentage (30%–38%) of overall contacts in the Gnutella network.

In addition to the already described approaches, Deschenes et al. [3] also had a running listener in their crawler, which was open to connections. During the membership protocol, the crawler would announce the IP/port of the listener and potentially solicit new connections from unknown peers.

Crawling in the above-described way is not limited to file-sharing networks like Gnutella or eDonkey [22] but has also been employed in VoIP programs such as Skype [7] or IPTV systems [8].

While the previously described crawlers all used a crawling strategy of only going through a queue of available peers, Saroiu et al. [19] followed a different strategy. Rather than crawling through the complete list of available peers, they limited the crawl time per snapshot to two minutes and then restarted the crawling process with the updated list of available peers. This, however, means that they only gather 25%–50% of the total population of peers in the system.

3.3 Bitcoin Crawler

In more recent times the most prominent peer-to-peer network that has been analysed with the use of crawlers has been Bitcoin. The measurement studies had different focuses, from the basic makeup of the network [4,16] to analysis of information propagation [2,10] and inference of topology [14,17] while others focused on deanonymisation of participants [1,11].

Similar to previous peer-to-peer crawlers, these crawlers utilised progressive crawling through available peers. For this purpose, either existing Bitcoin clients were extended for logging capabilities or dedicated crawler software was developed [11,13]. However, to our knowledge, most utilise a single instance approach rather than a distributed manager/worker architecture, and like most previous efforts they stick to a simple crawling strategy contacting all available nodes in each snapshot run.

Some monitors are publicly available such as the KIT DSN Bitcoin monitor[2] and Bitcoinstats.com, yet to our knowledge of the popular currently running monitors only Bitnodes publishes their code[3].

4 Design Considerations

4.1 Architecture

The Bitcoin crawler developed for this project was written as a multi-processing Python3 application. It utilises non-blocking, asynchronous I/O in order to maximise the number of concurrent connection attempts. In order to further increase crawling capacity for a single snapshot, we chose to implement a manager/worker architecture as pictured in Fig. 4. The crawler system, therefore, consists of a central coordinator instance and a variable number of crawling instances. The initial resolution of seed nodes, which are the entry point into the peer-to-peer network, is done at each individual crawling instance, in order to reduce geographic bias in domain resolution. However, all results of address requests are sent to the central coordinator, which then, in turn, gives out new tasks for the crawlers, so a single snapshot is created. Each crawling instance has a local cache of already discovered nodes. The cache minimises network traffic as these nodes do not have to be communicated to the coordinator again. Without the use of such a cache, the central coordinator would quickly become a bottleneck as after the initial discovery in the network. Only new nodes or nodes which had a change in status (online/offline) are sent to the coordinator. The workers themselves are made up of two different subsystems - the active participant in the neighbour discovery protocol and the passive listener. The active connector makes address requests to other peers while at the same time announcing the IP/port of the passive listener. Through this, the addresses of our crawlers are propagated throughout the network. These announcements can lead to unsolicited connection attempts by potentially previously unknown, especially new, peers in the network.

[2] dsn.tm.kit.edu/bitcoin.
[3] github.com/ayeowch/bitnodes.

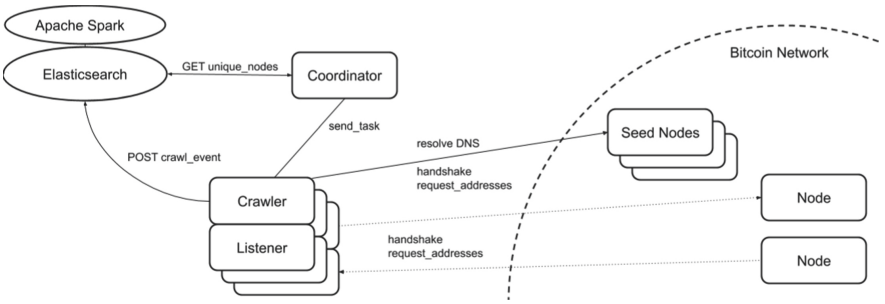

Fig. 4. Architecture overview

4.2 Bogon Filtering

A first approach to improve the hit rate of our crawler is to employ Bogon filtering. The filtering of private or reserved subnets as well as IP space that has not been assigned to an ISP yet is a common practice for firewalls. These are all addresses that should not be reachable on the internet. We utilise bogon IP lists provided by Team Cymru[4]. In regular operation, these should rarely be propagated by Bitcoin peers. However there can be scenarios where they are nonetheless propagated, which we will explain in the following. The Bitcoin reference implementation categorises peers in two categories, *good* and *terrible*. When an address request is received, a random selection of addresses is selected for the answer, but *terrible* nodes are not included. Four different checks characterise these *terrible* nodes: a timestamp more than 10 min in the future, not seen in 30 days, never responded after three attempts or seven separate failures during a week. Even after these checks, there can be instances where unreachable node addresses (i.e. Bogons) get propagated, such as very new addresses (addresses known less than a minute are exempt from the *terrible* rating) or a node might be reachable in a private network and therefore not receive a *terrible* rating yet be unreachable for our crawler from the internet.

4.3 Status Checking

Although there might be some Bogons in the Bitcoin network peer databases, it is much more likely that we encounter valid but unreachable IPs. Network address translation often used for consumer internet connections, firewalls or peers that simply left the network and do not run the peer-to-peer application in question anymore, are typical examples. However, these peers might be only temporarily unreachable, which is why we will have to periodically check their status in subsequent snapshots in order to keep an accurate picture of the network. In our experiments, of the reachable peers most were reachable during our first connection attempt. However, a third of peers that we found to be online at some

[4] team-cymru.org.

point during crawling only came online after we started our crawl. Interestingly the number of IPs that re-entered the network after an offline time was relatively low with 3%. We suspect that most residential setups which are more likely to enter and exit the network regularly are already not reachable due to NATs. In Fig. 5 the offline time of peers that either re-entered or came online during our crawl can be observed.

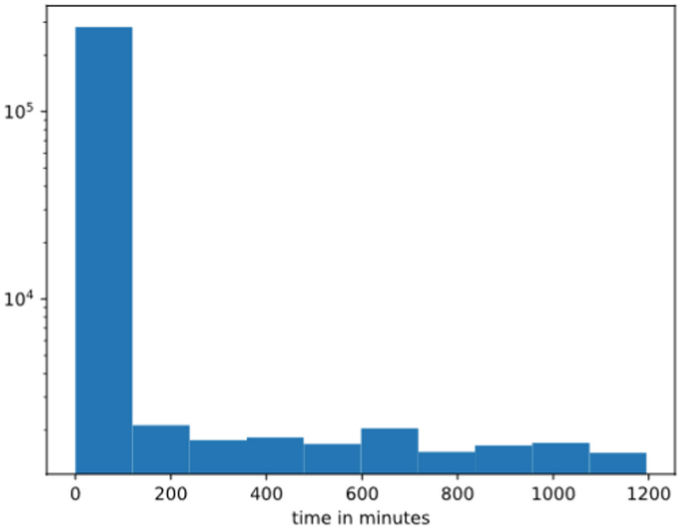

Fig. 5. Histogram of time until successful connection establishment

In order to further improve the performance of our crawler, we will use a kind of "back-off" on our status checks. This is modeled after the observed "dark" times of peers in our measurement. For this purpose, the central coordinator keeps track of the offline time for each participant. The longer a peer has been offline, the less likely it is that it will come back and be reachable. Therefore the coordinator will also be less likely to assign an IP to a crawler if said IP has been offline for a longer time, minimising likely unsuccessful connection attempts. This is implemented by comparing the duration since the *last_checked* date with the *offline_time* at the last check in combination with a backoff factor. In effect it is an implementation of exponential backoff, multiplicatively decreasing the rate at which a node is retried. However we added an upper limit on the decrease so that crawlers hit nodes at least once a day.

4.4 Optimal Timeout

As mentioned before there is a large portion of the network that will never be reachable for our crawlers. This includes already departed peers as well as peers that are unreachable behind a NAT or similar middleboxes. In these cases, the typical behaviour would be to wait for the *tcp_syn_retries* of the kernel to timeout and detect a failed connection establishment attempt. With the default settings of most Linux distributions, an initial connection timeout takes around 45 s. In order to reduce the required timeout, we measure the behaviour of alive peers to find the optimal trade-off between speed and completeness of the crawl. ZMap has shown that a stateless approach to scanning can be highly beneficial to the performance, however we can't employ such techniques as we need to not only perform a SYN-ACK scan but rather interact with the probed host on the application layer. NMap, as reference for a stateful high performance scanner, has different timing templates ranging from *"paranoid"* to *"insane"*. These can be applied depending on the desired intensity of the scan for the use-case. In order to choose an appropriate timing for our use-case we performed network measurement. In Fig. 6(a) this behaviour of alive peers can be observed. The time in seconds before a connection is established is shown as a boxplot. The median time is at 73 ms while the average is around 110 ms with a max of 380 ms. In order to capture the majority of attempts but not sacrificing performance in order to capture all outliers, we chose the timeout for the initial connection establishment as trade-off at 500 ms. This corresponds to the *aggressive* timing template of general NMap scans.

In order to improve the performance not only for unreachable peers but reachable peers as well, in addition to the *tcp_syn_retries*, we consider the TCP connection timeout of already established connections. If there is "silence on the wire", it could take a standard Linux host around 2 h to terminate the connection unless otherwise instructed. Finding a fitting timeout, therefore, is quite beneficial. In contrast to the *tcp_syn_retries* timeout, our idle timeout is considerably longer as it is not dominated by latency but rather the processing/queue at the client of the opposite peer. In Fig. 6(b)/(c) our measured duration of request responses can be seen. In (b) only the minimum measurements of each node are considered while (c) is a representation of all measurements. It shows that as a minimum half of all measurements are below 1.51 s while the upper half of measurements goes up to 13 s. When considering all measurements, however, we observe a considerably higher spread of delays. As these are only outliers and subsequent or previous requests were answered more quickly, we opt to choose the timeout according to the minimum latencies per node not overall. Hence the connection timeout for ongoing connections is set to 13 s.

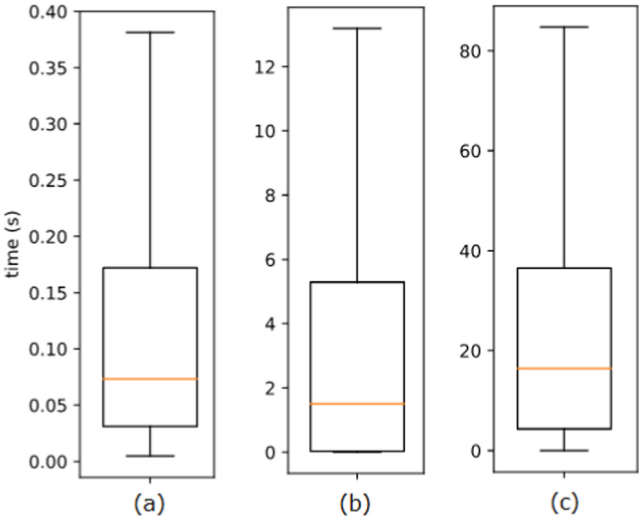

Fig. 6. (a) Time until successful TCP connection establishment (b) Minimum time per node until successful address request response (c) Distribution of all measurements for address request-response

NMap by default attempts to retransmit failed probes 10 times, however as Durumeric et al. have shown, even without retranmission of probes a large coverage of 98% can be achieved. As we will perform continuous crawling rounds we opted to not retransmit probes but rather reattempt the probe later according to the strategy described in the previous subsection.

5 Data Gathering

For the evaluation of the crawler improvements We collected data for 24 h on October 4th 2020. For this, we deployed the central coordinator in Berlin. The crawlers were geographically distributed using a mix of public cloud resources in Ireland, Ohio and Hong Kong as well as resources in Berlin. These regions (EU/NA/Asia) cover the vast majority of participants in the Bitcoin network. The crawlers were connected through IPv4 as well as IPv6 with roughly 250 address requests per second. During this time we connected to 251846 unique addresses of which 6643 were reachable.

5.1 Crawling Etiquette

As we are conducting active network measurements as opposed to less intrusive passive measurements, we have to consider some ethical questions. In order for participants in the network to be able to opt-out of being crawled we include a unique user-agent in our handshakes which includes information on how to

contact us. If we receive any complaints or requests for exclusion we add the addresses in question as exception to the hitlist (although no such request was made during our crawling). We use locally stored databases for OSINT (provided by MaxMind and TeamCymru) to avoid sending user data to third parties. After the OSINT has been gathered and our analysis is finished, we replace the IP which can be considered personally identifiable information (PII) with a pseudonym for storage.

5.2 Gathered Data

We will first give a rough overview of the gathered data, showing the functionality of the Bitcoin crawler as a network monitor. On the one hand we are replicating well known functionality such as enhancement of the gathered IPs with OSINT such as ASN, Geolocation and displaying handshake information such as user agent, protocol version and offered services. On the other hand we also offer latency data of our connection via ICMP, as TCPing and inside the Bitcoin protocol which approximates the processing delay of the remote peer. The difference between those latencies is shown in Fig. 7. Especially this extensive latency data which can be extended to include any port desired for a study can be used for interesting insights into to the composition of the network. As we have demonstrated with latency based proxy detection, as well as selective port scanning for proxy detection [15].

6 Evaluation

6.1 Increased Scan Rate

In order to compare the performance of our crawler and specifically the impact of our chosen timeout, we took a popular Bitcoin crawler that is open source as baseline[5]. Bitnodes chooses a single timeout for both the establishment as well as the idle connection, which by default is 30 s, compared with our 1s initial and 13s idle connection timeout. In Table 1 we show in which phase of the connection during our experiment run, connections actually timeout.

It can be observed that the vast majority of peers already fail during the initial TCP connection establishment. This means our improved timeout during this phase gives us a significant performance improvement. Due to the high number of peers behind NATs and firewalls unreachable to our crawler as well as the significant churn in the network, the vast majority (96.2%) of discovered nodes are never reachable and hence timeout. For this majority, our crawler overall (combining all phases and their probability) has an average 1.52 s timeout (94.93% lower compared to Bitnodes). Using the same resources (number of coroutines) this leads to a 94.7% faster scan rate.

[5] bitnodes.io.

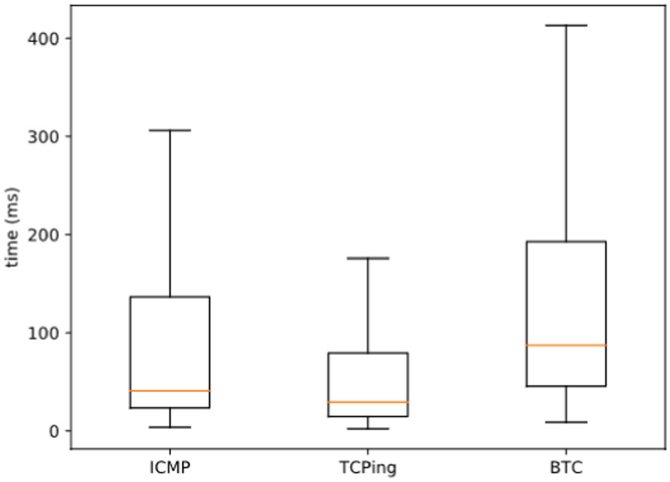

Fig. 7. Latency of reachable nodes measured via ICMP, SYN-ACK and Bitcoin TCP connection

Table 1. Phase of timeout in connection

Phase	Percentage
TCP connection establishment	95.6%
BTC handshake	2.4%
BTC address request	1.7%

6.2 Increased Hit Rate

One of our tactics to increase crawling speed is to not only have lower timeouts but avoid predictable timeouts altogether. The propagation of Bogons in the Bitcoin network seems to be minimal as during a 24 h crawl only 18 Bogons, mainly from private subnets, have been discovered. The increase in hit rate from Bogon filtering is, therefore, negligible. To evaluate the overall effectiveness of our measures, we compare the hit rate as well as discovered new peers of the typical consecutive snapshot strategy with our strategy that takes into account previous offline time. For this purpose, we ran our crawlers with and without the "back-off" strategy for at least 10 consecutive snapshots. The evaluation showed that the back-off strategy indeed increased the hit rate from 3.8% to 4.8% while the amount of discovered new peers stayed consistent with previous efforts. The crawl speed increase due to our back-off strategy was on average 9%.

7 Conclusion

We developed a new Bitcoin crawler that decreases blocking time through optimised timeouts as well as increased hit rate during consecutive network snapshots. The need for optimisation of crawling speed and resource usage has been

shown through the quantification of network characteristics that impact the accuracy of crawled snapshots. Through our evaluation, we quantified these improvements. Our Bitcoin network crawler increases scan rate by 94.7% compared to other publicly available Bitcoin monitors and increases the hit rate from roughly 3.8% to 4.8% due to taking into account the previous offline time of a node before including it in the crawl list. This new strategy also decreases the creation time of a snapshot by 9%.

Acknowledgements. This work has been funded by the German Federal Ministry of Education and Research (BMBF) under grant M534800. The responsibility for the content of this publication lies with the authors.

References

1. Biryukov, A., Khovratovich, D., Pustogarov, I.: Deanonymisation of clients in bitcoin P2P network. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 15–29 (2014)
2. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: IEEE P2P 2013 Proceedings, pp. 1–10. IEEE (2013)
3. Deschenes, D.G., Weber, S.D., Davison, B.D.: Crawling gnutella: Lessons learned. Technical report (2004)
4. Donet Donet, J.A., Pérez-Solà, C., Herrera-Joancomartí, J.: The bitcoin P2P network. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) FC 2014. LNCS, vol. 8438, pp. 87–102. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44774-1_7
5. Durumeric, Z., Wustrow, E., Halderman, J.A.: ZMap: fast internet-wide scanning and its security applications. In: 22nd {USENIX} Security Symposium ({USENIX} Security 13), pp. 605–620 (2013)
6. D'Acunto, L., Pouwelse, J., Sips, H.: A measurement of nat and firewall characteristics in peer-to-peer systems. In: Proceedings of the 15th ASCI Conference, vol. 5031, pp. 1–5. Citeseer (2009)
7. Guha, S., Daswani, N.: An experimental study of the skype peer-to-peer VoIP system. Cornell University, Technical report (2005)
8. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: A measurement study of a large-scale P2P IPTV system. IEEE Trans. Multimedia **9**(8), 1672–1687 (2007)
9. Imtiaz, M.A., Starobinski, D., Trachtenberg, A., Younis, N.: Churn in the bitcoin network: characterization and impact. In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 431–439. IEEE (2019)
10. Kanda, R., Shudo, K.: Estimation of data propagation time on the bitcoin network. In: Proceedings of the Asian Internet Engineering Conference, pp. 47–52 (2019)
11. Koshy, P., Koshy, D., McDaniel, P.: An analysis of anonymity in bitcoin using P2P network traffic. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 469–485. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_30
12. Lyon, G.F.: Nmap network scanning: The official Nmap project guide to network discovery and security scanning. Insecure (2009)
13. Maesa, D.D.F., Franceschi, M., Guidi, B., Ricci, L.: BITKER: a P2P kernel client for bitcoin. In: 2018 International Conference on High Performance Computing and Simulation (HPCS), pp. 130–137. IEEE (2018)

14. Miller, A., et al.: Discovering bitcoin's public topology and influential nodes (2015)
15. Mühle, A., Grüner, A., Meinel, C.: Characterising proxy usage in the bitcoin peer-to-peer network. In: International Conference on Distributed Computing and Networking 2021, pp. 176–185 (2021)
16. Neudecker, T.: Characterization of the bitcoin peer-to-peer network (2015–2018). Karlsruhe, Technical report, p. 1 (2019)
17. Neudecker, T., Andelfinger, P., Hartenstein, H.: Timing analysis for inferring the topology of the bitcoin peer-to-peer network. In: 2016 Internationl IEEE Conferences on Ubiquitous Intelligence and 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. IEEE (2016)
18. Ripeanu, M., Foster, I., Iamnitchi, A.: Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. arXiv preprint cs/0209028 (2002)
19. Saroiu, S., Gummadi, K.P., Gribble, S.D.: Measuring and analyzing the characteristics of Napster and Gnutella hosts. Multimedia Syst. **9**(2), 170–184 (2003)
20. Stutzbach, D., Rejaie, R.: Capturing accurate snapshots of the Gnutella network. In: Proceedings IEEE INFOCOM 2006, 25TH IEEE International Conference on Computer Communications, pp. 1–6. IEEE (2006)
21. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: Proceedings of the 6th ACM SIGCOMM Conference on Internet measurement, pp. 189–202 (2006)
22. Yang, J., Ma, H., Song, W., Cui, J., Zhou, C.: Crawling the Edonkey network. In: 2006 Fifth International Conference On grid and Cooperative Computing Workshops, pp. 133–136. IEEE (2006)