Robust P2P Connectivity Estimation for Permissionless Bitcoin Network

Hsiang-Jen Hong* Wenjun Fan* Simeon Wuthier* Jinoh Kim†
Xiaobo Zhou* C. Edward Chow* Sang-Yoon Chang*
* University of Colorado Colorado Springs, Colorado Springs CO, USA
{hhong, wfan, swuthier, xzhou, cchow, schang2}@uccs.edu
† Texas A&M University-Commerce, Commerce TX, USA
jinoh.kim@tamuc.edu

2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS) | 978-1-6654-1494-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/IWQOS52092.2021.9521287

***Abstract*—Blockchain relies on the underlying peer-to-peer (p2p) networking to broadcast and get up-to-date on the blocks and transactions. It is therefore imperative to have high p2p connectivity for the quality of the blockchain system operations. High p2p networking connectivity ensures that a peer node is connected to multiple other peers providing a diverse set of observers of the current state of the blockchain and transactions. However, in a permissionless blockchain network, using the peer identifiers—including the current approach of counting the number of distinct IP addresses and port numbers—can be ineffective in measuring the number of peer connections and estimating the networking connectivity. Such current approach is further challenged by the networking threats manipulating the identifiers. We build a robust estimation engine for the p2p networking connectivity by sensing and processing the p2p networking traffic. We implement a working Bitcoin prototype connected to the Bitcoin Mainnet to validate and improve our engine's performances and evaluate the estimation accuracy and cost efficiency of our estimation engine.**

***Index Terms*—Bitcoin, Blockchain, Cryptocurrency, Peer-to-peer networking, Permissionless, Reliability, Robustness**

I. Introduction

Cryptocurrency such as Bitcoin replaces a centralized authority/bank with a distributed ledger to store and process the financial transactions to provide anonymous and censorless financial transactions. Enabling such properties are the distributed consensus protocol and networking designed to operate in *permissionless* environments (which lacks the registration or the identity-based control while still achieving fairness across the cryptocurrency participants). The distributed consensus protocol is based on proof of work (PoW) and measures the fairness based on the computational power of the participants, called miners, as opposed to their number of identities. For example, in PoW, hundred miners each of which has a computational power of x H/s is designed to have the same probability of finding the block as one miner having a computational power of $100x$ H/s. Such design is the main innovation by Nakamoto in his seminal Bitcoin paper [1], in which he reinforced the permissionless design and the anonymity by recommending new identifiers/accounts for every transaction. Such permissionless design motivates our work in this paper, as we challenge the effectiveness of the current approach based on counting distinct identities and build peer-connectivity estimation without relying on identities but on the networking traffic.

9978-0-7381-3207-5/21/$31.00 ©2021 IEEE

Underlying blockchain and the distributed consensus protocol is the broadcasting network based on peer-to-peer (p2p) networking. The p2p network provides the block and the transaction information to the peers, and the health of the p2p network determines when they receive such information. If a miner has an unhealthy network of peers (limited connectivity) and does not receive the blocks on time, then it mines on outdated blocks wasting its resources on blocks without rewards. Newer cryptocurrencies such as Ethereum provide partial block rewards to PoW on outdated blocks [2]–[4] to provide greater fairness in networking health and yield some rewards to the peers with lower p2p connectivity. However, the newer cryptocurrencies also rely on the p2p networking and require healthy networking connections for their operations. The connectivity is therefore critical in the mining operations and in ensuring the fairness between the miners.

In this work, we propose a connectivity estimation engine, which provides accurate estimation and is effective even in the cryptocurrencies' permissionless and anonymous environments. Our work provides estimations of the peer connectivity even when there is a spoofing or Sybil attacker present (manipulating the peer connections based on false identifiers/IP). Such threats driven by malicious peers represent the worse-case scenario where the *Legacy* approach of counting the network-layer identities, e.g., IP addresses and port numbers, becomes ineffective in measuring the health peer connection. Our work is motivated to address the above issue. To this end, we build the connectivity estimation engine without relying on identities but based on analyzing the networking traffic and behaviors for estimating the health peer-connection. To build robustness against the threats manipulating the estimation and the training, we include an Outlier Detection (OD) in our proposed engine to detect outliers (§ II-B2) before making the viable estimation decisions. Without OD, such threats can affect the estimation result; even if the threat is intelligent enough to forgo OD, OD can mitigate and tamper the impact of the threats. We also show the effectiveness of OD with prototype-based experiments (§ V-B). To validate and evaluate our peer connectivity estimation engine, we build an implementation prototype on a Bitcoin node and test it with the Bitcoin Mainnet networking (§ III-A).

978-1-6654-1494-4/21/$31.00 ©2021 IEEE

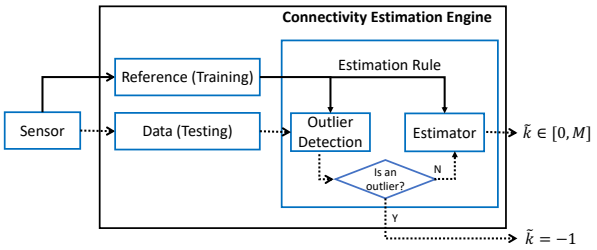


Fig. 1: The Operational Flow

II. Our Connectivity Estimation Engine

Our estimation engine is motivated by the permissionless and trustless properties and is designed for cryptocurrencies. Even though our connectivity estimation engine is generally applicable to other permissionless blockchain networks relying on its underlying p2p networking, our prototyping and experiments focus on the Bitcoin cryptocurrency.

A. Operational Flow

Figure 1 illustrates the operational flow of our connectivity estimation engine. It provides an overview of deploying and utilizing our estimation engine on a Bitcoin node for users. Users should first deploy a Sensor unit for collecting the Bitcoin message traffic and transmitting the data to the estimation engine. The data collected from the Sensor unit first go through the training path (solid line). The collected data transmit to the Reference unit for building the training references. Sequentially, the Reference unit passes the references to the Estimation Rule unit as the significant references for Outlier Detection and Estimator. After finishing the training, the data collected by the Sensor will go through the testing path (dotted line). The Data unit processes the collected data by storing the network-monitor outputs for testing. The result is passed to the Outlier Detection unit to filter out the outlier that is different from our training reference. Assume the finalized estimation result of health peer connection is $\widetilde{k}$. Then, the engine will directly set $\widetilde{k}$ as -1. Such data will not go to the Estimator unit. Only the data that successfully passes the Outlier Detection examination is meant to be estimated for its health peer connection. Assume that the maximum number of peer connections set by a Bitcoin node is M. The finalized result of $\widetilde{k}$ will be located within [0, M]. In summary, users can use our engine to estimate $\tilde{k}$ accurately and filter out the abnormal traffic by comparing it with the reference we built for that specification.

B. Design of Connectivity Estimation Engine

We propose the connectivity estimation engine to estimate the number of health peer entities connected. Our engine does not use the identifier-based information but rather the networking traffic information (which does not distinguish between the networking streams from different nodes but aggregate them for the networking behavior). Our estimation engine is built on the hypothesis that the networking behavior changes with respect to the number of connected peer entities (rather than their identities) based on which we design our engine and test it using a working prototype on the real-world Bitcoin network. The engine is constituted by three components: Per-Parameter Processing (PPP), Outlier Detection (OD), and Estimation Aggregator (EA). We introduce these components in detail in the following paragraphs.

1) Per-Parameter Processing (PPP): Before going to the operation of PPP, we first investigate the networking traffic parameters that we want to measure (monitor) for our estimation. Among the networking parameters, we focus on those that can better inform the peer connectivity health. More specifically, we focus on the *Traffic Analyses Parameters* and the *Packet (Bitcoin-specific) Analyses Parameters*.

- *Traffic Analyses Parameters: n and s.* The networking traffic parameters are those general networking traffic information which popularly uses in the traffic analyses: the count (packet) rate, n, i.e., the number of packet arrivals per time; the aggregate networking size rate, s, i.e., the bandwidth information.
- *Packet (Bitcoin-specific) Analyses Parameters: λ and n_{m_i}.* To better capture more useful information for estimation, we further analyze the Bitcoin-specific packets. In this regard, we focus on two parameters: the relative frequency distribution across the bitcoin p2p networking message types, λ, and the per-message count rate, n_{m_i}.

There are 26 message types, i.e. $m_i, 1 \leq i \leq 26$, used in the Bitcoin protocol for exchanging information between peers by now, including those for block and transaction propagation. Our estimation uses the above networking parameters given from the networking sensing. We train those parameters to set the references for estimation. For the frequency distribution, we introduce the reference of frequency distribution when k peers are connected to the Bitcoin node, Λ_k. Similarly, we denote the count rate reference, the size rate reference, the per-message count rate references given k peer connections as $\bar{n}_k$ and $\bar{s}_k$, $\bar{n}_{m_i,k}$ respectively. The above references are derived and computed by the average values from the training samples n_k, s_k, λ_k and $n_{m_i,k}$. After obtaining the references, the PPP is ready to go. We separate the cases into PPP for EA and PPP for OD. PPP for EA aims to output the estimation result $\widetilde{k}_x$ based on per-parameter x only. In contrast, PPP for OD aims to provide an outlier score $\widetilde{o}_x$ based on x.

PPP for EA: The estimation involves comparing between the real-time testing (with a time window size of T_{test}) and the training references (with a time window size of T_{train}), which comparison method depends on the parameters themselves. For the packet count rate n and the networking size rate s, the estimation is based on the rate magnitude differences. Specifically, we will compute the difference between the testing n and different $\bar{n}_k$ where $\bar{n}_k$ is computed by the average count rate of k peer connection's training data. The PPP identifies the $v = \arg\min_k(|n - \bar{n}_k|)$. If the $n - \bar{n}_v \geq 0$, then $\widetilde{k}_n = v + \frac{n-\bar{n}_v}{\bar{n}_{v+1}-\bar{n}_v}$. Otherwise, $\widetilde{k}_n = v - \frac{\bar{n}_v-n}{\bar{n}_v-\bar{n}_{v-1}}$. $\widetilde{k}_n$ is defined as the $\widetilde{k}$ when using the n parameter only. A similar approach is also applicable to the estimation based on size rate for obtaining $\widetilde{k}_s$ and the per message count rate for obtaining $\widetilde{k}_{n_{m_i}}$. This approach makes sense if the

reference value has an ascending order when k increases. We examine it and observe that the reference value of n and s have an ascending trend when k increases. However, we do not include those results due to space constraints. For n_{m_i}, we examine those messages that have this trend and choose two messages that have the highest capabilities in distinguishing different peer connections, which will be further discussed in §II-B3. For the frequency distribution λ, we use the correlation coefficient denoted as ρ for comparing λ and Λ_k where Λ_k is the frequency distribution across the different Bitcoin message types. The correlation magnitude of ρ is between 0 and 1. The higher the ρ is, the greater the similarity (between the monitored testing data and the reference). We denote ρ_k as the computed correlation coefficient by λ and Λ_k. Then, the estimation decides the subject Bitcoin node has $\widetilde{k}_\lambda = \arg\max_k(\rho_k)$ peers.

PPP for OD: We also define a per-parameter outlier score computed based on x as $\widetilde{o}_x$ to indicate the level of outlierness of a sample. In OD, we use $\widetilde{o}_x$ for making the finalized decision whether a sample should be treated as an outlier or not by using all $\widetilde{o}_x$. We think that the outlierness should be checked by comparing whether the testing sample is indeed similar to its respective training references. Note that if $\widetilde{k}_x = z$, only the training reference given by z peer connections will be utilized to detect the outlier. Suppose the testing sample is highly different from the respective training references. In that case, it should be treated as an outlier and unable to go into the finalized estimation, i.e., EA. We propose using the ratio-based approach. Our engine checks the ratio between the testing sample and its respective training reference. Take n as instance, if n is the count rate of a testing sample with $\widetilde{k}_n = z$, the outlier score $\widetilde{o}_n$ will be conducted by $\frac{n}{\bar{n}_z}$. Similar operations can also adapt to the s and n_{m_i}. One might notice that λ is not suitable for the ratio-based approach. Hence, we directly utilize ρ because the value reveals the similarity between the testing and the training. If a testing sample with $\widetilde{k}_\lambda = z$, we directly set $\widetilde{o}_\lambda = 1 - \rho_z$ to indicate its outlierness.

2) Outlier Detection (OD): As we mentioned earlier in § II-B1, $\widetilde{o}_x$ is computed by the ratio-based approach. Here, we focus on how to aggregate the $\widetilde{o}_x$ among different parameters x and make the final decision. Note that only those testing data pass the examination of outlier detection is viable to be further estimated by EA. We utilize logical-OR for the aggregation of $\widetilde{o}_x$ scores to output $\widetilde{o}$. The output $\widetilde{o}$ is a binary factor. If $\widetilde{o}$ is 1, it represents that the testing data is an outlier, and $\widetilde{k}$ is immediately set to -1. If $\widetilde{o}$ is 0, it represents that the testing data is not an outlier and will be sent to EA for estimation. We use the threshold control in identifying the outliers. To be more specific, if $\widetilde{o}_x$ is not located within a ratio range β=[β_l, β_h], then a binary factor b_x representing the outlier result is set to one. Otherwise, $b_x = 0$ Additionally, a threshold control is also required for $\widetilde{o}_\lambda$. We denote the threshold set for $\widetilde{o}_\lambda$ as α. If a testing sample with $\widetilde{o}_\lambda > \alpha$, then $b_\lambda = 1$. Finally, the finalized output is conducted by a logical-OR operation, i.e., $\widetilde{o} = b_n \vee b_s \vee b_\lambda \vee b_{n_{m_i}}$.

3) Estimation Aggregator (EA): After filtering the outlier traffic by OD, EA aggregates the results of $\widetilde{k}_x$ from PPP and produces the final estimation decision. We use a weighted function between the networking parameters and make the estimation $\widetilde{k}$ where $x \in \{n, s, \lambda, n_{m_i}\}$. Since there are 26 message types, i is an integer located within [1, 26].

$$\widetilde{k} = \sum_x w_x \cdot \widetilde{k}_x, \quad x \in \{n, s, \lambda, n_{m_i}\}, i \in \{i | 1 \le i \le 26\} \quad (1)$$

$$\approx \sum_x w_x \cdot \widetilde{k}_x, \quad x \in \{n, s, \lambda, n_{\text{ADDR}}, n_{\text{PONG}}\} \quad (2)$$

We actually do not select all the message types for computing n_{m_i} for our estimation but only ADDR and PONG messages because they have higher capabilities in distinguishing different peer connections. The finalized equation used in our engine is summarized in Equation (2). We identify a message type with a higher capability to distinguish different peer connections by Kolmogorov–Smirnov test (KS test). KS test is a standard technique in statistical analysis [5] for identifying whether two samples' distribution are quite different or not. The KS test examines the shape and distance of the two samples' CDF distribution. It outputs a p value to measure how similar they are. Higher p where $p \in [0, 1]$ indicates that they have more similar distributions. To use the KS test in determining the distinguishing capability, we compute the CDF for distribution of n_{m_i} under $k, 1 \ge k \ge 10$ peer connections. m_i is a message type that has a monotonically increasing trend on n_{m_i} while varying k. For each consecutive CDF distributions of n_{m_i} under k and $k+1$, we compute the respective p value. Then, we compute the average p value for these nine p values computed for one specific message type m_i. The lower the average p value obtained by a message type m_i, the higher the distinguishing capability the message m_i has. Based on our obtained result, PING, PONG, VERACK, and ADDR, have the four lowest average p values. However, the VERACK will only be transmitted on the Bitcoin peer connection establishment stage, i.e., one-time transmission. Hence, it is not suitable for selecting it as a parameter for estimation. In addition, for PING and PONG, since PONG is the responded message type that reveals more information about the peer's liveness, we only select PONG. All the weights in Equation (2) are summed up to be one. We will further analyze the impact of the weight control in the estimation performances in § IV. While the actual number of peers connected is k, the estimation engine decides that there are $\widetilde{k}$ peer connections. Here, $\widetilde{k}$ is not constrained to integer. However, one can decide how to utilize the $\widetilde{k}$, such as computing the rounding number of $\widetilde{k}$ and assuming the rounding number the finalized estimated peer connection.

III. Bitcoin Peer and Connectivity Estimation Engine Implementation

A. Prototype Implementations

We implement our Connectivity Estimation Engine prototype on a Bitcoin node running the Bitcoin Core (software version Satoshi 0.18.0 and protocol version 70015). Our Bitcoin node implementations are based on virtual machines (VM)

that have the same specifications using Linux Mint 19.2 Tina (64-bit) with four processors and 6144 MB memory. We run in private mode (have up to 10 connections and have limited exposure to the Internet since it cannot be discovered by the other peers from the Internet). The estimation for the public Bitcoin nodes with 125 connections is left for future work. In fact, similar operations and architecture can also be applied to the public Bitcoin nodes. However, the public nodes that have additional inbound peer connection have more unstable traffic and require a more comprehensive analysis. In this work, we focus on private nodes to better control the peer connections and the networking traffic generated by our own peers. We implement two Bitcoin nodes, including the node hosting our estimation engine (X) and another attacking node (A). The operation of A will be further demonstrated in § V-A. In this section, we focus on no attack case for X. From our Bitcoin peer X, we collect both the training data and the testing dataset from the real-world Bitcoin Mainnet by controlling the peer connections of X. For training, X connects to k number of peers ($0 \leq k \leq 10$) on the Mainnet, and every training dataset selects random k peers on the Internet due to the randomness in the selected peers' networking conditions.

IV. Evaluation Results

This section shows the experimental results about the estimation performance and the cost-efficiency of our engine. Our evaluation analysis builds on the prototype implementation. We collect 1000 minutes of data for each dataset and divides them into 800 minutes for training the references and 200 minutes for the testing purpose. For the 200 minutes of testing data, we set $T_{test} = 20$ minutes. In other words, we use ten testing samples with each dataset. The ratio between training and testing with 80% training and 20% testing is popularly used [6]. Our work also aligns with this setting. All datasets are separate and non-overlapping in time.

A. Estimation Metrics

We use two metrics, Mean Square Error (MSE) and ϵ-Tolerance Accuracy, for evaluating the estimation performance. Both MSE and ϵ-Tolerance Accuracy measure the estimation accuracy; however, MSE is continuous, while ϵ-Tolerance Accuracy is based on the discrete decision. More specifically, MSE provides the raw error without discretization, while ϵ-Tolerance Accuracy makes the discrete decision for $\widetilde{k}$ and derives the error based on $|\widetilde{k} - k|$. Because each testing sample results in a decision that is either correct or erroneous, the $Accuracy = 1 - \Pr[Error]$ where Pr[.] is the empirical probability. We define the $Error$ events to incorporate a tolerance level of ϵ and call such accuracy "ϵ-Tolerance Accuracy." More specifically, our estimation engine allows the estimation to be off by $\pm\epsilon$, and $Error$ occurs if $|\widetilde{k} - k| > \epsilon$.

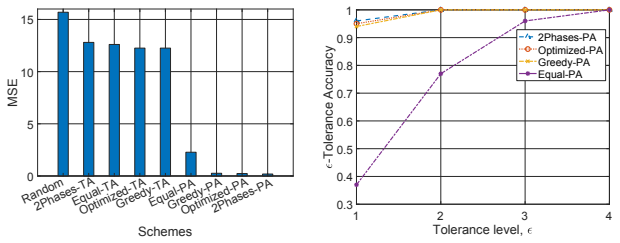
B. Weight Control Schemes

Now, we introduce four weight control schemes we proposed for EA. The first scheme is the *Equal* scheme using equal weights. The second one is the *Greedy* scheme. We examine the mean of $\widetilde{k_x}$ to see which x provides the closest mean to k and select it as a primary factor. We examine the weight setting of w_x from 0 to 1 with a granularity of 0.01. The remaining weights are equally distributed to all the other parameters. For instance, assume that w_n is the first primary factor. We examine which weight of w_n can conduct the lowest MSE; we directly fix the weight of w_n with the lowest MSE. And then, we select the next primary factor and make a similar approach until we fix all the weights. The third scheme is called *Optimized* scheme. We use an exhaustive search for this scheme to examine the best weight setting obtaining the lowest MSE. The efficiency of obtaining the optimized weight setting by using exhaustive search should be exponentially increased when the number of used parameters increases. However, we only use five parameters in our scheme to obtain the result within a reasonable time.

According to our observation, the parameters which are useful for the lower peer connections might not still be beneficial for the higher peer connections. Therefore, we proposed the last *2Phases* scheme. As the name indicates, the 2Phases scheme has two phases: i) the EA first classifies the networking traffic being tested between low connectivity vs. high connectivity and ii) then controls the parameter weights to optimize the performances in that case. Specifically, we separate them into the relatively lower peer connectivity case ($1 \leq k \leq 5$) and higher peer connectivity case ($6 \leq k \leq 10$). $k = 0$ produces zero networking traffic. The high connectivity case presents a greater challenge because there are greater peer connections, which will yield more randomness on the Bitcoin traffic. For the first phase, we utilize Λ for the binary classification. To be more specific, we compute the frequency distribution references for lower and higher peer connectivity, which are denoted as Λ_{low} and Λ_{high}, respectively. These references are computed by averaging the k cases, that is, $\Lambda_{low} = (\sum_{i=1}^{i=5} \Lambda_i)/5$ and $\Lambda_{high} = (\sum_{i=6}^{i=10} \Lambda_i)/5$. We then compare the frequency distribution of the testing data, Λ_{test} with Λ_{low} and Λ_{high} using the correlation coefficient ρ, similarly to how we utilize the frequency distribution references for the peer connectivity estimation. The first-phase classification decides the low connectivity vs. the high connectivity by choosing the case with the higher correlation ρ. For the second phase, we optimize the weight control given whether it is the low-connectivity or the high-connectivity. Note that we only use the optimized approach for the two-phase to obtain a better estimation accuracy, i.e., the lower MSE.

C. Estimation Performance

All the schemes associated with different weight settings can all have the traffic analysis (TA) approach and the additional packet analyses (PA) approach, respectively. We denote them as follows: Equal-TA, Equal-PA, Greedy-TA, Greedy-PA, Optimized-TA, Optimized-PA, 2Phases-TA, and 2Phases-PA. We measure all the above schemes to reveal the estimation performances using different schemes. As shown in Figure 2(a), the accuracy performance is significantly improved for PA compared to TA. To be more specific, the MSE for the



(a) Average MSE on All Schemes (b) ϵ-Tolerance Accuracy on PA

Fig. 2: Estimation Performance ("Random" corresponds to randomly guessing $\tilde{k}$ from a uniform distribution)

PA approach was 2.2764 or lower, depending on the weight-control algorithm. In contrast, the TA approach has MSE performance greater than 12.2537. Undoubtedly, the Random scheme makes a blind guess at k and therefore performs the worst among all schemes. The Greedy-TA and Optimized-TA have the same MSE. This is because there are only two parameters in the TA approach. The operation for finding the best weight setting of greedy and optimized schemes is the same. One interesting observation is that 2Phases-TA does not have a better result than Greedy-TA and Optimized-TA. This is because these two parameters are not especially sensitive to low/high peer connections. On the contrary, it makes the estimation results worse due to the deficiency of the comprehensive views for minimizing the error. However, in the PA approaches, the additional used parameters are sensitive to low/high peer connections. In this regard, the 2Phases approach allows us to obtain a better estimation performance.

Figure 2(b) compares the performances of the estimation schemes while varying ϵ from 1 to 4. We mainly focus on PA schemes due to their superior performance compared to the TA schemes. All schemes increase in the accuracy performances when there is greater tolerance (increasing ϵ). The 2Phases-PA scheme obtains a 97% 1-Tolerance Accuracy which is better than other accuracy results obtained by all other schemes. The Equal scheme does not attempt to optimize the weight but uses equal weights. Therefore, it performs the worst accuracy performance with only 37% in 1-Tolerance Accuracy.

D. Cost Analysis

We measure the cost of the time spent during the weight control algorithms and the time spent on testing for Greedy-PA, Optimized-PA, 2Phases-PA in Table. I. Even though the costs on Optimized-PA, 2Phases-PA are significantly higher than the greedy in the weight control stage, the estimation accuracy for these two still outperform than the greedy one. In other words, this is a trade-off between the estimation performance and the pre-processing cost (i.e., Weight control). If one wants to obtain a better estimation performance, the cost of the exhaustive search might still be acceptable. More importantly, such costs will only spend during the weight control stage. We also check the cost spent during testing. Because the 2Phases-PA needs additional operation in distinguishing the low/high peer connections for the testing samples, it has a relatively high cost compared with the other two schemes. In summary, we recommend using 2Phases-PA and Greedy-PA for different purposes. If one aims for a better estimation accuracy, 2Phases-PA is the best solution with the lowest MSE. In contrast, if one aims for a decent estimation accuracy with a less computational overhead, Greedy-PA provides a more lightweight solution.

Estimation Aggregator:	**Greedy-PA**	**Optimized-PA**	**2Phases-PA**
Weight control algorithm	0.0474s	102.1649s	102.5622s
Testing	0.0787s	0.0825s	0.1057s

TABLE I: Cost Analysis among Different Schemes

V. Outlier Detection for Countering Threats

We analyze the robustness of our connectivity estimation engine in countering the worse-case networking failures caused by the security threats. To simulate the failures caused by security attacks, we simulate and prototype two attacks: the man-in-the-middle attack (which is related to the recent networking threats in blockchain such as Eclipse attack [7], routing attack [8], and Erebus attack [9]. Specifically, our man-in-the-middle attack builds on and uses Sybil attack for greater threat impact. It also has the same effect as the Eclipse attack from the receiver's perspective since the attacker controls all the traffic seen by the victim) and the DoS attack. In our experiment, the threshold for $\widetilde{o}_\lambda$, α, is set to 0.5. The ratio range β is set to $\beta = [0.5, 1.5]$ for OD.

A. Prototyping

We prototype a man-in-the-middle attack where an attacker A located between the victim X and the Bitcoin Mainnet. All traffic communicated between X and the Bitcoin Mainnet can be manipulated by A. Because we try to emulate the worst case for X where k and $\tilde{k}$ has a large difference, the man-in-the-middle attacker in our prototyping generates M (the maximum peer connection of X) fake identities (Sybil IPs) and links to the victim. Note that X only links to these M peer connections controlled by the attacker A. If X tries to link to some other peers, the attacker can manipulate the traffic and make it unachievable. The above case should be the most harmful case to the victim. In this scenario, if the victim only utilizes the peer connection information provided by the $Legacy$ approach, the value should be totally different from the health peer connection k. For ease of our exposition, we denote the estimation provided by the $Legacy$ approach as $\widetilde{k}_{Legacy}$. In such a case, k should be zero and $\widetilde{k}_{Legacy} = M$ where M=10 in our experiment. In addition, the attacker will not relay any packet from the Mainnet to the victim (to harm the victim by restricting its information from the Mainnet) and will not respond any message to X. However, X will send the PING messages to all of its peers (controlled by A) to check the liveness of the connections. The connection will be stopped by X if A does not respond by the PONG message. In such a case, if one of the connections is disconnected by X, A will immediately establish a new connection to X with another Sybil IP to maintain the worst case scenario.

We also prototype the PING DoS Attack especially using the Bitcoin PING messages. The attacker A establish one

connection to the victim X. After the connection is established, the attacker keeps sending Bitcoin PING messages to the victim, trying to flood the victim's memory or CPU usage. In this scenario, X can still accept the information from the Mainnet through other healthy peer connections. However, the PING DoS flooding occupies the bandwidth of X and causes slower healthy traffic received from the healthy peers.

B. Countering man-in-the-middle and DoS attack

We collect 20 samples with $T_{test} = 20$ for both man-in-the-middle attack and DoS attack, respectively, in this prototyping. Our OD successfully detects the security threats among all 40 samples and sets the finalized $\widetilde{k}$ for these samples as -1. In contrast, if the victim uses the *Legacy* approach, $\widetilde{k}_{Legacy} = 10$ for all samples, i.e., the victim is not able to detect the threats. We therefore incorporate OD into our estimation engine to build robustness against threats causing abnormal outliers.

VI. Related Work

For dynamic distributed network systems such as p2p network, the connectivity reliability is of paramount importance as a metric for the network system. In [10], Xiong and Liu presented PeerTrust, a reputation-based trust supporting framework, to estimate peers' trustworthiness. Hao et al. [11] presented a trust-enhanced blockchain p2p topology that accelerates the transmission rate and retains transmission reliability. However, they focus on the reliability from the network-view, whereas our work focuses on the host-view. The permissionless property can make dynamic behavior even further and decrease the reliability. Thus, we propose the connectivity estimation engine to address the reliability concern.

Building on Sybil, Heilman et al. [7] propose a threat on the bitcoin network where the attacker manipulates the victim's peer connections for controlling the information flow. Such peer-connection control enables the attacker to control the block/transaction information delivery to the victim, and further launch selfish mining [12] or double-spending attack [7]. Our work is robust against such identity control threats because we estimate the connectivity from the peer itself and use the networking traffic information (as opposed to the finer-granular identifier-based or packet-based information). There are some other network-based attacks in the bitcoin network, such as routing attack [8], partitioning attack [9], and mempool flooding [13], DDoS attack by spam transactions [14]. Those attacks enabled by changing the normal traffic behavior, such as increasing the per-message count rate, can possibly be detected by our estimation engine. Our work mainly focuses on the reliability issue raised by permissionless blockchain networks by accurately estimating the health peer connections and filtering out abnormal traffic.

VII. Conclusion

The permissionless blockchains such as those used for Bitcoin and other cryptocurrencies forgo the reliance on a centralized entity for providing the trust/registration and enable anonymous and censorless transactions. However, the permissionless nature of cryptocurrencies challenges the reliance on peer identities because the peers can and are even encouraged to use multiple identities for anonymity. In such environment, the prior legacy approach for measuring and estimating the peer connectivity by using the peer identifier information can become ineffective, e.g., against identity manipulations. In this paper, we propose a robust connectivity estimation engine by analyzing the networking traffic and behaviors. Based on our implementations, we recommend the 2Phases-PA scheme to optimize the accuracy performance as it achieves 97% estimation accuracy with a tolerance of one peer connection and has the MSE of 0.18789. Moreover, we show the effectiveness of our outlier detection (OD), especially against the networking threats, and use it as a part of the estimation engine to enhance the connectivity estimation.

Acknowledgment

This research was supported in part by Colorado State Bill 18-086 and by the National Science Foundation under Grant No. 1922410.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," 2014. [Online]. Available: http://gavwood.com/paper.pdf

[3] F. Ritz and A. Zugenmaier, "The impact of uncle rewards on selfish mining in ethereum," in *Proceedings of the 2018 IEEE EuroS&P Workshops*, April 2018, pp. 50–57.

[4] S.-Y. Chang, Y. Park, S. Wuthier, and C.-W. Chen, "Uncle-block attack: Blockchain mining threat beyond block withholding for rational and uncooperative miners," in *Proceedings of the 17th International Conference on Applied Cryptography and network Security*, 2019, pp. 241–258.

[5] H. W. L., "On the kolmogorov-smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.

[6] Y. Ding, Y. Du, Y. Hu, Z. Liu, L. Wang, K. Ross, and A. Ghose, "Broadcast yourself: Understanding youtube uploaders," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, New York, NY, USA, 2011, p. 361–370.

[7] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proceedings of 24th USENIX Security Symposium (USENIX Security 15)*, Aug 2015, pp. 129–144.

[8] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *Proceedings of the 2017 IEEE S&P*, May 2017, pp. 375–392.

[9] M. Tran, I. Choi, G. Moon, A. V. Vu, and M. Kang, "A stealthier partitioning attack against bitcoin peer-to-peer network," in *Proceedings of the 2020 IEEE S&P*, may 2020, pp. 515–530.

[10] L. Xiong and L. Liu, "Peertrust: supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

[11] W. Hao, J. Zeng, X. Dai, J. Xiao, Q. Hua, H. Chen, K. Li, and H. Jin, "Towards a trust-enhanced blockchain p2p topology for enabling fast and reliable broadcast," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 904–917, 2020.

[12] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *Proceedings of the 2016 IEEE EuroS&P*, March 2016, pp. 305–320.

[13] M. Saad, L. Njilla, C. Kamhoua, J. Kim, D. Nyang, and A. Mohaisen, "Mempool optimization for defending against ddos attacks in pow-based blockchain systems," in *Proceedings of the 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 285–292.

[14] J. Zhang, Y. Cheng, X. Deng, B. Wang, J. Xie, Y. Yang, and M. Zhang, "Preventing spread of spam transactions in blockchain by reputation," in *Proceedings of the 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, 2020, pp. 1–6.