Computer Networks 219 (2022) 109436

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Robust P2P networking connectivity estimation engine for permissionless Bitcoin cryptocurrency

Hsiang-Jen Hong [a,*], Wenjun Fan [b], Simeon Wuthier [a], Jinoh Kim [c], C. Edward Chow [a], Xiaobo Zhou [a], Sang-Yoon Chang [a,*]

[a] Department of Computer Science, University of Colorado at Colorado Springs, Colorado Springs, CO, USA
[b] Department of Communications and Networking, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China
[c] Department of Computer Science, Texas A&M University, Commerce, TX, USA

ARTICLE INFO

Keywords:
Bitcoin
Blockchain
Cryptocurrency
Peer-to-peer networking
Permissionless
Robustness

ABSTRACT

Blockchain relies on the underlying peer-to-peer (P2P) networking to broadcast and get up-to-date on the blocks and transactions. Because of the blockchain operations' reliance on the information provided by P2P networking, it is imperative to have high P2P connectivity for the quality of the blockchain system operations and performances. 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 Bitcoin cryptocurrency 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 identities. We build a robust estimation engine for the P2P networking connectivity by sensing and processing the P2P networking traffic. We take a systematic approach to study our engine and analyze the followings: the different components of the connectivity estimation engine and how they affect the accuracy performances, the role and the effectiveness of an outlier detection to enhance the connectivity estimation, and the engine's interplay with the Bitcoin protocol. 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 connectivity estimation engine. Our results show that our scheme effectively counters the identity-manipulations threats, achieves 96.4% estimation accuracy with a tolerance of one peer connection, and is lightweight in the overheads in the mining rate, thus making it appropriate for the miner deployment.

1. Introduction

Cryptocurrencies based on blockchain technology have become much more popular in the modern financial market. Trading between different currencies is also feasible by using cryptocurrency exchange schemes [1,2]. 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 [3], in which he reinforced the permissionless design and the anonymity by recommending new identifiers/accounts for every transaction. Such permissionless design also 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 which does not rely on identities but on the networking traffic and behaviors.

Underlying blockchain and the distributed consensus protocol is the broadcasting network based on peer-to-peer (P2P) networking. The

* Corresponding authors.
E-mail addresses: hhong@uccs.edu (H.-J. Hong), schang2@uccs.edu (S.-Y. Chang).

https://doi.org/10.1016/j.comnet.2022.109436
Received 22 July 2022; Received in revised form 24 October 2022; Accepted 27 October 2022
Available online 8 November 2022

networking channels are critical because it provides the necessary information for the miner peers' blockchain and consensus operations. More specifically, the P2P network provides the block and the transaction information to the PoW-participating 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. To mitigate such wastage issue and increase fairness for the peers with lower P2P connectivity (e.g., developing regions), newer cryptocurrencies such as Ethereum provide partial block rewards to PoW on outdated blocks [4–6]. While these mechanisms highlight the importance of healthy connections for cryptocurrencies, instead of the mitigation based on partially compensating for mining on the outdated blocks due to poor connectivity, we take a different approach and our goal is to inform the networking connectivity.

In this work, we propose a peer 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 peer connection health. In order to be robust against such identity manipulations, we design and build our connectivity estimation engine. We focus on statistical analysis rather than the machine learning approach. Although machine learning algorithms have been used in similar domains such as anomaly detection [7,8], the machine learning-based approaches require significant computational resources and affect the mining performance. We thus adopt the statistical analysis approach to make the estimation engine lightweight and easily deployed. While the statistical and machine learning approaches are both data-driven, statistical analysis is more interested in uncovering the characteristics of variables and their relationships. For instance, we use our preliminary analyses conducted in Section 4.2 to limit the selected parameters used for estimation.

For designing our connectivity estimation engine based on a statistical analysis approach, we first identify the useful networking-sensor parameters for our connectivity estimation engine by analyzing the networking traffic. After deciding the parameters, the training phase algorithm is twofold. First, the algorithm computes the average values for each parameter under k connections as the training references. Second, the algorithm uses k-fold cross-validation to find the best weight setting with the smallest estimation error. After getting the best weight setting of the estimation equation, the testing phase algorithm takes the training references, the best weight vector, and the real-time testing data as inputs. The algorithm computes the per-parameter estimation result by comparing the real-time testing and the training references. To further build robustness against the threats, we include an Outlier Detection (OD) to detect outliers (Section 3.2.3) before making viable estimation decisions. Specifically, the testing phase algorithm computes the outlier scores for testing data. If an outlier is detected, it will be filtered out and will not be used for the final estimation aggregation. In contrast, if the testing data is not an outlier, a final estimation for aggregating the per-parameter estimation result outputs the final estimation result.

The contributions of this paper can be summarized as follows.

- We design our connectivity estimation engine for permissionless Bitcoin network. We build the engine on an active Bitcoin node and improve our engine at multiple components and levels.
- We examine the useful networking-sensor parameters for our connectivity estimation engine by analyzing the networking traffic. In addition to processing the aggregate networking traffic such as packet counts or received bandwidth (*Traffic Analyses*), we explore distinguishing the incoming networking according to the Bitcoin applications and message types (*Packet Analyses*) (Section 3.2.1).
- Our work involves the improvement and optimization in the aggregation and the weight control of the per-parameter decisions. (Section 5.2). We also analyze the tradeoff between the time complexity in the execution and the accuracy performance of the estimation schemes (Section 5.4.2).
- To validate and evaluate our peer connectivity estimation engine, we build an implementation prototype on an active Bitcoin node and test it with the real Bitcoin Mainnet networking (Section 4.1). To analyze the robustness of our engine in countering the worst-case networking failures, we prototype three attacks, man-in-the middle attack, PING DoS attack, and spam transactions flooding (Section 6.1). The result shows that the proposed OD schemes is robust against those attacks. Based on our prototype study, we recommend 2Phases for connection estimation scheme and LoOP (Local Outlier Probabilities)-Weighted for outlier detection.

The rest of the paper is organized as follows: Section 2 describes the problem statement and the motivation. Section 3 discusses the detailed design of our proposed connectivity estimation engine. We implement our connectivity estimation engine and conduct preliminary analyses for parameter selection in Section 4. Section 5 presents the experimental results on estimation accuracy and cost-efficiency. In Section 6, we prototype three attacks and show the effectiveness of our engine in countering those networking threats. In Section 7, we review related work in the literature. Section 8 draws conclusions with future research direction.

2. Problem statement & Motivation

As we describe in Section 1, the permissionless property in the cryptocurrency application (for anonymous transactions) challenges the identity-control-based trust in its underlying networking. Due to the permissionless property, a networking peer in permissionless blockchain network can use multiple identities. In fact, generating multiple identities per peer-entity is encouraged in such environments where the recommendation is to generate and use new identities for every transaction for anonymity [3]. A peer can also generate Sybil-like IP addresses and/or use the Tor network to anonymize traffic and complement the application-layer anonymity requirements. In the case of a networking peer with malicious intentions, the peer can manipulate the peer identities for conducting Sybil threat (constructing multiple bogus identities) and spoofing threat (masquerading as another existing peer) [9,10].

The current Bitcoin node only tracks its peer information based on the peer's IP address the port number. Specifically, the number of peers is estimated based on simply counting the identifiers in the IP address and the port number (the number of records/connections in the peer table). We call such a scheme used for the current bitcoin network the *Legacy* approach. Unfortunately, the *Legacy* approach is vulnerable against the aforementioned threats manipulating the peer identities. Our work is motivated to address the above issue. To this end, we build a connectivity estimation engine robust against such identity manipulations in permissionless Bitcoin network.

We build the connectivity estimation engine without relying on identities but based on analyzing the networking traffic and behaviors for estimating the peer-connection health. We additionally involve an outlier detection to determine the viability of connectivity estimation and only estimate the peer connections when there is no outlier detected. We highlight the effectiveness challenge of the *Legacy* approach against the identity-manipulating networking threats and demonstrate the robustness of our connectivity estimation engine in Section 6.2.

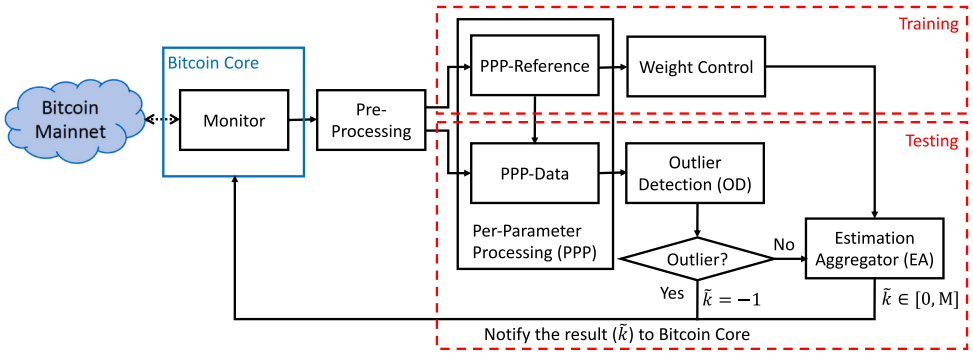


Fig. 1. An overview of the connectivity estimation engine.

3. Our connectivity estimation engine

Our connectivity estimation engine is oblivious to the peer identifiers and is more robust to attacks or identity obfuscation, as described in Section 2. Our 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.

3.1. Overview of connectivity estimation engine

In this section, we give an architecture overview of our connectivity estimation engine illustrated in Fig. 1. It provides an overview of deploying and utilizing our connectivity estimation engine on a Bitcoin node connecting to the Bitcoin Mainnet with real-world Bitcoin traffic.

- **Monitor:** The Monitor is a built-in function module of the Bitcoin Core (software version Satoshi 0.18.0 and protocol version 70015) to monitor and collect all Bitcoin traffic from peers. The Bitcoin traffic provides information like the timestamps, the number of peers, the identity of those connected peers, the accumulated number of messages for each Bitcoin message type at that timestamp, the total bytes received aggregated by Bitcoin message types at that timestamp, etc. Therefore, our connectivity estimation engine can utilize this module to collect the data for our estimation purpose. For obtaining the required Bitcoin traffic used for training, we collect the Bitcoin traffic under different $1 \leq k \leq M$ peer connections, where M is the maximum number of peer connections set by the Bitcoin node.
- **Pre-processing:** In Pre-processing, we use a script to compute and retrieve the focused parameters of our connectivity estimation engine from the collected data given by Monitor. We discuss the focused parameters in detail in Section 3.2.1. The reason why those parameters are selected for our estimation is provided and analyzed in Section 4.2. After Pre-processing, the data with T_{train} size will first go through the training phase (the upper red rectangle identified in Fig. 1.) Once the training is finished, the following testing data with T_{test} size go through the testing phase (the lower red rectangle identified in Fig. 1.)
- **PPP-Reference:** PPP-Reference is about building the training references of the focused parameters based on Per-Parameter Processing (PPP). Specifically, we compute the average values for each parameter under k connections as the training references during the training phase. PPP-Reference first outputs the training references to the Weight Control for selecting the best weight setting with the minimum estimation error. PPP-Reference also outputs the training references to PPP-Data involved in the testing phase to compare with the testing data and obtain the per-parameter estimation result ($\widetilde{k}_x$ where x is the focused parameter).
- **Weight Control:** During the Weight Control, we adopt the k-fold cross-validation to find the best weight setting with the minimum estimation error among different weight control algorithms. To be more specific, the original training data will be split into k partitions. The validation will sequentially take one partition as the testing sample, whereas the remaining partitions are training data. The Weight Control outputs the best weight setting to Estimation Aggregator (EA) to aggregate all $\widetilde{k}_x$ into the finalized estimation ($\tilde{k}$).
- **PPP-Data**: PPP-Data processes each testing data with T_{test} size. For each testing data, PPP-Data computes the average values on those focused parameters as the testing values. PPP-Data compares the difference between the testing values and the training references and make a per-parameter estimation decision ($\widetilde{k}_x$). In addition, PPP-Data also outputs per-parameter outlier scores ($\widetilde{o}_x$) based on two different approaches, the 1.5 interquartile range rule and the Local Outlier Probability approach.
- **Outlier Detection (OD)**: We called an outlier as an observation that appears to be inconsistent with other observations in the training dataset. To remedy the bias raised by possible existing abnormal peers, we collect training data from peers and keep changing the peer sets during the data collection stage in Monitor. In OD, we aggregate the per-parameter outlier scores ($\widetilde{o}_x$) to the finalized outlier score $\widetilde{o}$. If the $\widetilde{o}$ is set to 1 (i.e., the data is detected as an outlier), The engine will directly set $\widetilde{k} = -1$ and notify Bitcoin Core about the outlier. Such data will not go to the Estimation Aggregator (EA). Only the data that successfully passes the OD examination is meant to be further estimated.
- **Estimation Aggregator (EA)** After filtering out the outliers, EA aggregates the per-parameter estimation $\widetilde{k}_x$ into the finalized estimation result $\widetilde{k}$ based on a weighting function where the weight setting is provided by the Weight Control. The finalized result of $\widetilde{k}$ is located within $[0, M]$ and will be sent to the Bitcoin Core.

Table 1 summarizes the variable notations used to describe and analyze our connectivity estimation engine. Our connectivity estimation engine is generally applicable to permissionless blockchain networks (built upon P2P network) and other P2P network applications because all the parameters we use are obtainable in P2P networks. With different applications, settings, and instances, our connectivity estimation engine would be tuned differently with different values assigned for the reference parameters and thresholds. We also align our training and testing so that they are adjacent in time/days as opposed to being further apart in days so that the training is the most effective for the estimation testing. Although generally applicable, in this work, we focus on using our connectivity estimation engine for the most popular Bitcoin network.

Table 1
Variable notations.

Variables	Meaning
k	The actual number of peer connections
$\widetilde{k}$	Finalized result estimated by the connectivity estimation engine
M	The maximum peer connections set by a Bitcoin node
n	Aggregate message (packet) count rate
s	Networking size rate
λ	Frequency distribution between the Bitcoin message types (for testing)
n_{m_i}	Message (packet) count rate of m_i message
$\bar{n}_k$	Count rate reference given k peer connections
$\bar{s}_k$	Size rate reference given k peer connections
Λ_k	Frequency distribution reference given k peer connections
$\bar{n}_{m_i,k}$	Message count rate reference for m_i given k peer connections
ρ	Correlation coefficient
ρ_k	Correlation coefficient between Λ_k and λ
$\widetilde{k}_x$	Estimation decision for k by parameter x
$\widetilde{o}_x$	Outlier score based on parameter x
$\widetilde{o}$	Finalized outlier score
w_x	Weight for the parameter x
T_{train}	Time duration of the training data
T_{test}	Time duration of the testing data
p	Result of KS test for measuring the similarity
ϵ	Tolerance level
$\overrightarrow{w}$	A weight vector $\overrightarrow{w} = [w_n, w_s, w_\lambda, w_{n_{\text{ADDR}}}, w_{n_{\text{PONG}}}]$
$\widetilde{k}_{Legacy}$	Number of peer connection obtained by the Legacy approach

*All the networking traffic is filtered to only the Bitcoin message traffic.

Table 2
Bitcoin message types.

ID	Message type	Notion
m_1	VERSION	HandShake Initiation
m_2	VERACK	Response to the VERSION
m_3	ADDR	Send a max of 1000 IP addresses
m_4	INV	Send a max of 50 000 trans./blocks
m_5	GETDATA	Response to INV
m_6	GETHEADERS	Request up to 2000 headers
m_7	TX	Response to GETDATA with trans.
m_8	HEADERS	Response to GETHEADERS
m_9	BLOCK	Response to GETDATA
m_{10}	PING	Request PONG
m_{11}	PONG	Response to PING
m_{12}	NOTFOUND	Response to GETDATA
m_{13}	REJECT	Inform the reject of a message
m_{14}	GETADDR	Request IP addresses
m_{15}	MEMPOOL	Request transactions from peers
m_{16}	SENDHEADERS	Request headers
m_{17}	FEEFILTER	Ignores trans less than 8bytes
m_{18}	SENDCMPCT	Request a compact block
m_{19}	CMPCTBLOCK	Response to SENDCMPCT
m_{20}	GETBLOCKTXN	Request a BLOCKTXN message
m_{21}	BLOCKTXN	Response to GETBLOCKTXN
m_{22}	MERKLEBLOCK	For transmitting a merkle block
m_{23}	GETBLOCKS	Request up to 500 blocks
m_{24}	FILTERLOAD	Set the filter that filters INV
m_{25}	FILTERADD	Add a bloom filter
m_{26}	FILTERCLEAR	Clear bloom filter

3.2. Design of connectivity estimation engine

We introduce the design of our connectivity estimation engine for the Bitcoin node in detail. Our proposed connectivity estimation engine estimates the number of healthy peer entities connected. As motivated in Section 2, 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 connectivity 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. Our work focuses on the connectivity estimation, which can inform other active measures to improve the connectivity; such active measures are left for future work.

3.2.1. The selection of estimation parameters

Before going to the operation of PPP, we need to first investigate the networking traffic parameters that we want to use for our estimation. Among the networking parameters, we focus on those that can better inform the peer connectivity. 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 message 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 (summarized in Table 2), 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 which can be obtained from the Monitor and be retrieved and computed by the Pre-processing. To analyses the relations between the parameter and the number of peer connection, we conduct the preliminary analyses in Section 4.2 to better establish the selection of the parameters as well as the insights on the parameter choices and how they affect the estimation engine design and performance.

3.2.2. Per-Parameter Processing (PPP)

PPP-reference. 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 data.

PPP-data for outputting $\widetilde{k}_x$. For outputting the per-parameter estimation result $\widetilde{k}_x$ based on x, we need to make a comparison between the real-time testing and the training references. For conducting the comparison based on $x \in \{n, s, n_{m_i}\}$, we utilize the interpolation approach this is because the reference values have ascending orders when k increase (which have been examined in Section 4.2). For example, if the testing n is located between $[\bar{n}_k, \bar{n}_{k+1}]$, $\widetilde{k}_n = k + \frac{n-\bar{n}_k}{\bar{n}_{k+1}-\bar{n}_k}$. 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 correlation coefficient computed by λ and Λ_k. Then, the estimation decides $\widetilde{k}_\lambda = \text{argmax}_k(\rho_k)$.

PPP-data for outputting $\widetilde{o}_x$. Similar to $\widetilde{k}_x$, the engine computes a per-parameter outlier score ($\widetilde{o}_x$) based on parameter x. For obtaining $\widetilde{o}_x$, the engine also compare the testing with the training reference. To be more specific, if the testing sample is estimated as $\widetilde{k}_x = z$, only the training reference given by z peer connections will be utilized for computing the $\widetilde{o}_x$. We first propose using the standard 1.5 interquartile range rule [11] to obtain $\widetilde{o}_x$. Specifically, any testing data that is more

Algorithm 1 Training Phase

Require: Bitcoin training data with T_{train} size
Ensure: $\bar{n}_k$, $\bar{s}_k$, Λ_k, and $\bar{n}_{m_i,k}$, $1 \le k \le M$, $\overrightarrow{w}$
1: **for** $1 \le k \le M$ **do**
2: Compute the references: $\bar{n}_k$, $\bar{s}_k$, Λ_k, and $\bar{n}_{m_i,k}$
3: Assign $\overrightarrow{w}$ outputted by the weight control algorithm with smallest MSE

than 1.5 interquartile (Q3-Q1, obtained by the training data) below the lower quartile (Q1) or more than 1.5 interquartile above the upper quartile (Q3) are considered as outliers. If the testing sample based on x is detected as outlier, than $\widetilde{o}_x = 1$. Otherwise, $\widetilde{o}_x = 0$. Since λ is not suitable for taking the quartile based approach, we directly utilize ρ because the value has already revealed the similarity between the testing and the training reference. We directly set $\widetilde{o}_\lambda = 1$ if $\rho_z < 0.5$. Otherwise, $\widetilde{o}_\lambda = 0$. The above quartile-based approach gives a binary result of $\widetilde{o}_x$. However, $\widetilde{o}_x$ can be a value reflecting the level of outlierness and not have to be a binary factor. To this end, we propose using the Local Outlier Probabilities (LoOP) [12]. LoOP attempts to tackle the dilemma that other methods face about choosing the suitable threshold for outlier scores to distinguish outliers by outputting the result as outlier probability in [0,1]. LoOP is categorized as a nearest neighbor-based outlier detection method. We demonstrate the operation of LoOP as follows. It first computes the density of a testing data point. It assumes that the testing data point is at the center of its neighborhoods. The distance to its k-nearest neighbors (denoted as a set S) follows a half-Gaussian distribution. With the above assumption, a probabilistic set distance is defined to estimate density around the data point based on S. To normalize the density concerning the average density based on S, a Probabilistic Local Outlier Factor (PLOF) is defined to compute the respective ratio. To achieve a normalization making the scaling of PLOF independent of the particular data distribution, the aggregate value nPLOF is obtained during PLOF computation. Finally, a Gaussian error function is applied to map the result constituted by PLOF and nPLOF into the LoOP score, which can be directly interpretable as a probability of a data object being an outlier. To make use of LoOP, we take the respective training data based on z peer connections and split it to $\frac{T_{train}}{T_{test}}$ data segments. Then, we compute the corresponding data points based on the average value of the data segments. Those data points generated based on training data are added to S and the average value of the testing data is treated as the testing data point to compute the respective outlier probability which is assigned as $\widetilde{o}_x \in [0, 1]$.

3.2.3. Outlier Detection (OD)

After obtaining $\widetilde{o}_x$ for all parameters x, we aggregate all $\widetilde{o}_x$ results into the final result $\widetilde{o}$. For quartile-based approach, we use the logical-OR operation. That is, $\widetilde{o} = \vee_{x \in \{n,s,\lambda,n_{m_i}\}} \widetilde{o}_x$. In other word, if the testing sample is treated as an outlier based one of the parameters, then we directly treat this sample as an outlier. For LoOP, we propose using a weighted function to compute the finalize $\widetilde{o}$ to provide flexibility in setting the weights on the score function for detecting more potential threats. That is, a finalized weighted outlier score equation is decided by $\widetilde{o} = 1$ if and only if $\sum_x w_x \cdot \widetilde{o}_x > 0.5$ where $x \in \{n, s, \lambda, n_{m_i}\}$. Otherwise, $\widetilde{o} = 0$.

3.2.4. Estimation Aggregator (EA)

After filtering the outlier traffic by OD, EA aggregates the results of $\widetilde{k}_x$ 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].

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

Algorithm 2 Testing Phase

Require: Bitcoin testing data with T_{test} size, the training references: $\bar{n}_k$, $\bar{s}_k$, Λ_k, and $\bar{n}_{m_i,k}$, $1 \le k \le M$, $\overrightarrow{w}$
Ensure: $\widetilde{k}$
1: Retrieve n, s, λ, n_{m_i} of the testing sample
2: Compute $\widetilde{k}_x$, $x \in \{n, s, n_{m_i}\}$ ▷ interpolation approach
3: Compute $\widetilde{k}_\lambda$ ▷ highest ρ approach
4: Compute $\widetilde{o}_x$, $x \in \{n, s, \lambda, n_{m_i}\}$, and the finalized $\widetilde{o}$
5: **if** $\widetilde{o} == 1$ **then** ▷ Filter out the outlier based on OD
6: $\widetilde{k} = -1$ ▷ Detect as outlier and notify the Bitcoin node
7: **else** ▷ Go to EA for the final estimation
8: $\widetilde{k} = \sum_x w_x \cdot \widetilde{k}_x$

Table 3
VM's specification for running a Bitcoin node.

Operating system	Linux Mint 19.2 Tina (64-bit)
Processor number	4
Memory	6144 MB
Network adapter	Intel PRO/1000 MT Desktop

All the above weights are summed up to be one. We actually do not select all the message types for computing n_{m_i} for our estimation but only ADDR and PONG messages. We use ADDR and PONG messages because they have higher capabilities in distinguishing different peer connections. In addition, PONG is directly used for connection maintenance and therefore provides good representations of the connectivity, as we explain in greater details in Section 4.2.3. As we mentioned in Section 3.1, the best weight setting for the function is provided by the output of Weight Control. We will further analyze the impact of the Weight Control in estimation performances in Section 5. While the actual number of peers connected is k, the connectivity 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. By default, we assume that $\widetilde{k}_x$ as floating numbers and the finalized $\widetilde{k}$ will be rounded into integer. We will further discuss the effect of the granularity control in Section 5.3. To help better understand the training and testing phases for our connectivity estimation engine, we summarize the operations in pseudo-codes in Algorithms 1 and 2.

4. Prototyping: Bitcoin peer and connectivity estimation engine implementation

4.1. Prototype implementations

We implement our connectivity estimation engine 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) with equal specifications (which are described in Table 3) and 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 connections bring additional challenges (more unstable traffic). More comprehensive analysis with additional beneficial parameters is needed for obtaining an accurate estimation result. In this work, we focus on private nodes to better control the peer connections (for training) and the networking traffic generated by our own peers (e.g., so that our attack simulations are contained within our lab environment).

We implement two Bitcoin nodes, including the one hosting our estimation engine (X) and another attacking node (A). The operation

of the attacking node A will be further demonstrated in Section 6.1. In this section, we focus on no attack case for X. From our Bitcoin peer X, we collect both the training data and the normal testing dataset from the real-world Bitcoin Mainnet by controlling the peer connections of X. For training, X connects to k number of peers on the Mainnet where $0 \leq k \leq 10$ and every training dataset selects random k peers on the Internet due to the randomness in the selected peers' networking conditions.

4.2. Preliminary analyses

As described in Section 3.2, we keep track of the networking traffic parameters: message (packet) count rate, n, networking size rate, s (both of which are aggregated over all the message types), and the Bitcoin-specific message parameters: the relative frequency distribution of the message types, λ (distinguishing between the message types), and the per message count rate, $n_{m_i}, 1 \leq i \leq 26$. This section focuses on the observation and the analysis for these parameters.

4.2.1. Analyses for n and s

In Fig. 2, we examine the cumulative distribution function (CDF) of n and s to see whether the distribution shifts when k increases. The CDF distributions of n and s are plotted based on the training datasets. More specifically, the time duration of training datasets we collected for each k is 800 min, i.e. $T_{train} = 800$ min. Undoubtedly, if the distribution of the parameter shifts significantly, it represents that this parameter might be more beneficial for distinguishing different k. The result shows that the CDF of n shifts as k grows. However, the CDF of s does not have a clear shifting pattern. By this observation, one can imagine that the s parameter might have a negative impact on the peer connectivity estimation. Note that $k = 0$ case is omitted because it corresponds to having no connectivity and yields $n = 0$ packets per second and $s = 0$ bytes per second.

4.2.2. Analyses for λ

The frequency distribution λ keeps track of the frequencies/occurrences across the different message types of the monitored data. For the λ from the testing dataset, the per-parameter estimation in turn compares it with the Λ_k from the training references using the correlation coefficient, ρ. Here, T_{train} is still set as 800 min, and T_{test} is set as 20 min. In addition, for each testing dataset with different peer connections, we have 10 testing samples for each of them. Fig. 3 presents a scenario using the testing samples from one peer connection and ten peers connection and the correlation coefficient ρ with the training reference datasets (Λ_k where $1 \leq k \leq 10$). The vertical axis plots the ρ between the testing and the training while the horizontal axis varies the training reference with respect to the connected number of peers k. To be more specific, for each point in Fig. 3, the ρ value is the average ρ computed among 10 testing samples and the compared reference. For clarification, the datasets between testing and training are collected at separate, non-overlapping time periods, i.e., the training and the testing are separate and do not overlap in time. When testing one-peer ($\lambda | k = 1$), ρ is greatest and has a decreasing trend as the k increases from the Λ_k in training. Even though the trend is not monotonically decreasing, the $\widetilde{k_\lambda}$ is still equal to k because the corresponding ρ has the highest value. If the estimation uses Λ only and makes its estimation decision based on that, then $\widetilde{k} = \widetilde{k}_\lambda = 1$ for the one-peer testing. However, for the ten-peers testing case ($\lambda | k = 10$), the relation between k and the ρ is not clear. The estimation for the ten-peer testing based on Λ can result in a poor estimation. Therefore, some other parameters are needed for obtaining a better estimation accuracy. In this regard, the selection of the message types for using the respective per message count rate is strongly required for this purpose and is introduced in the following section.

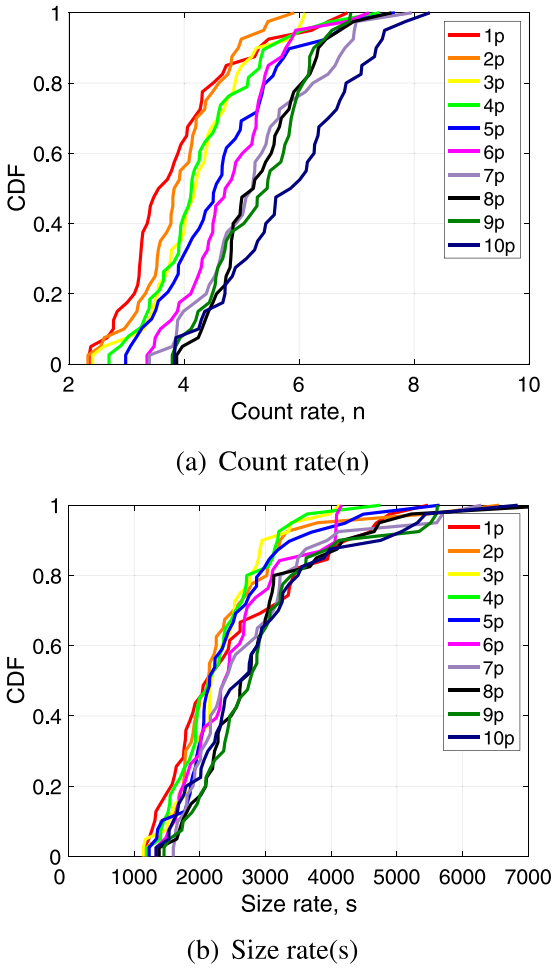


Fig. 2. CDF of n and s while varying k (from $k = 1$ to $k = 10$)

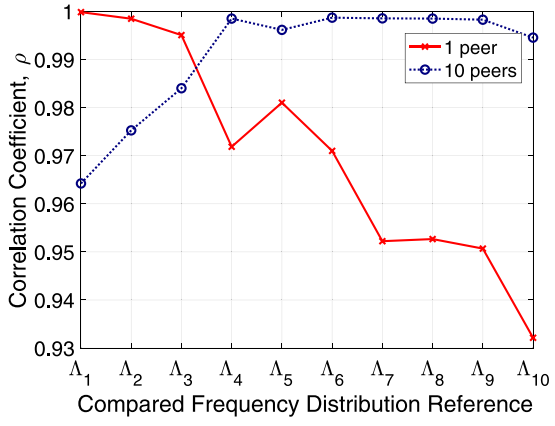


Fig. 3. Λ comparison.

4.2.3. Analyses for n_{m_i}

For analyzing which per message count rate n_{m_i} is more beneficial for our connectivity estimation engine, we first determine the message types m_i, which do not increase proportionally with k in principle and avoid using them in our estimation. Take the SENDHEADERS message as an example. The count rate will not proportionally increase with k because the rate is mainly controlled by whether the node's peers enable some techniques or not. The remaining m_i, whose packet counts are $\bar{n}_{m_i,k}$ given k peer connections. are plotted in Fig. 4. Based on the

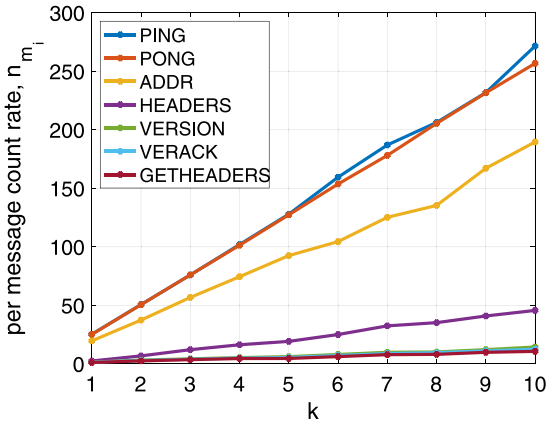


Fig. 4. m_i with ascending order of $\bar{n}_{m_i,k}$.

Table 4
KS test of the targeted message types.

Message type	$\bar{p}_{m_i}$
PONG	0.1482e−08
PING	0.4991e−08
VERACK	0.0311
ADDR	0.0905
VERSION	0.1565
GETHEADERS	0.2098
HEADERS	0.3223

results, we can observe that the per message count rate of PING, PONG and ADDR have bigger slopes than other message types.

To better select the message types for our estimation, we use the empirical distributions of the message-specific counts given k, $\bar{n}_{m_i,k}$ and apply to Kolmogorov–Smirnov test (KS test), which is a standard technique in statistical analysis [13,14] 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 and 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 be more specific, to use KS test, we compute the CDF for distribution $n_{m_i,k}, 1 \geq k \geq 10$ where m_i are those messages that have the monotonically increasing trend. For each consecutive pair ($n_{m_i,k}$ and $n_{m_i,k+1}, 1 \geq k \geq 9$) of the CDF distributions, we compute their p value, denoted as $p_{m_i,k,k+1}$. Then, we compute the average p value for this nine p values computed for one specific message type m_i denoted as $\bar{p}_{m_i}$. We summarize the $\bar{p}_{m_i}$ for those m_i that have the monotonically increasing trends in Table 4. According to the conducted $\bar{p}_{m_i}$ listed in Table 4, we only select PONG and ADDR as the message types that we want to utilize its per message count rate. As we stated earlier, the lower the $\bar{p}_{m_i}$ has, the more capability it has for distinguishing different peer connections. By observing the result, PING, PONG, and VERACK are the three smallest ones. 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. We then select ADDR, which is located in the fourth place as the parameter. In addition, for PING and PONG, since PONG is the responded message types which reveal more information about the peer's liveness, we only select PONG as a parameter. Other message types are filtered out for our estimation since they might obfuscate the estimation of peer connections. In this regard, we use the estimation equation listed in Eq. (2) for final estimation.

$$\widetilde{k} = \sum_{x} w_x \cdot \widetilde{k}_x, \quad x \in \{n, s, \lambda, n_{\text{ADDR}}, n_{\text{PONG}}\} \tag{2}$$

5. Evaluation results

This section shows the experimental results about the estimation performance and the cost-efficiency using our estimation engine. Our evaluation analysis builds on the prototype implementation and the preliminary analyses in Section 4. As mentioned in Section 4.1, we vary the number of connections and use fresh, randomly selected peers every 50 min to generate the datasets to remedy the bias raised by possible existing abnormal peers. The training and testing datasets are different with no overlap. Our experiment collects 1000 min of data for each k peer dataset and divides them into 800 min for training the references and 200 min for the testing purpose. For the 200 min testing data, we set $T_{test} = 20$ min (which is twice the expected period for Bitcoin block arrivals since Bitcoin blocks get mined every ten minutes in expectation by design). In other words, we use ten testing samples with each dataset. Our work follows the ratio of 80% training and 20% testing because it is popularly used in both statistical analysis and machine learning [15,16].

5.1. Estimation metrics

For evaluating the estimation performance, we use two different metrics, Mean Square Error (MSE) and ϵ-Tolerance Accuracy. Both MSE and ϵ-Tolerance Accuracy measure the estimation accuracy; however, MSE is continuous, while ϵ-Tolerance Accuracy is based on the discrete decision.

- **Mean Square Error (MSE):** The main metric used for evaluating the estimation performance of our proposed method is the Mean Square Error denoted as MSE, a common criterion for evaluating the quality of an estimation.
- **ϵ-Tolerance Accuracy:** We also define an ϵ-Tolerance Accuracy as an auxiliary metric for our estimation. Because each testing sample results in a decision which 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 $|\widetilde{k} - k| > \epsilon$. For example, if $\epsilon = 0$, then the estimation is correct only when $\widetilde{k} = k$ and, if $\epsilon = 1$, then the estimation can be off by ± 1 so that the estimation is correct when $|\widetilde{k} - k| \leq \epsilon = 1$. In our experiments, we focus on $\epsilon = 0$ and $\epsilon = 1$ cases because we already get very high accuracy performances using $\epsilon = 1$.

Undoubtedly, the MSE metric provides relatively richer information compared to ϵ-Tolerance Accuracy and does not need to identify the tolerance level (ϵ) beforehand. In view of this, we only examine the ϵ-Tolerance Accuracy for the schemes using the additional packet analyses (which outperforms the schemes only use the traffic analysis parameters) in Fig. 6(b). One might notice that the tolerance level ϵ used for is a design parameter for our estimation engine and depends on the cryptocurrency peer state in the randomness/variance in the P2P networking and the number of peer connections supported. For example, in default Bitcoin protocol, while a private node (such as the one used in our experiment) has up to 10 connections so that $k \in \{0, 1, 2, \ldots, 10\}$, a public node can have up to 125 peer connections so that $k \in \{0, 1, 2, \ldots, 125\}$. The greater the options for k, the greater the tolerance level, ϵ to have comparable accuracy performances. In our experiment, we mainly focus on the peer connectivity estimation for the private nodes. The analysis for the public nodes is left for future work. Notably, the private nodes occupy a large portion of the nodes in the entire Bitcoin network. Even though the vulnerability of private nodes is less than the public nodes, the victim node is still suffered from the threats of Sybil attack or other attacks that manipulates the victim's peer connections if the attacker is located within the same local area network.

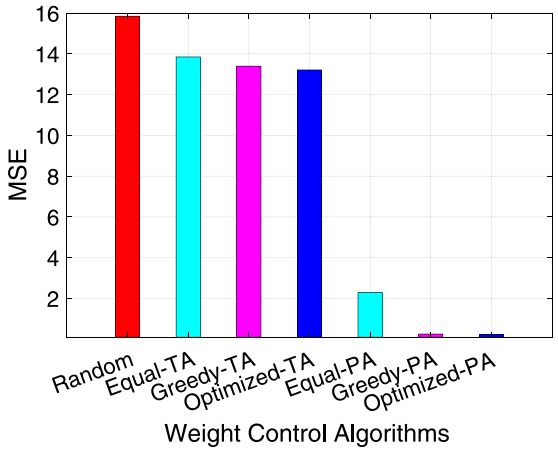


Fig. 5. Effect of the weight control.

5.2. Effect of the weight control

In this section, we analyze the effect of Weight Control (during the training phase) using different weight control algorithms proposed as follows.

The greedy approach: We first propose the greedy approach. For ease of our exposition, we denote a weight vector $\overrightarrow{w} = [w_n, w_s, w_\lambda, w_{n_{\text{ADDR}}}, w_{n_{\text{PONG}}}]$ to represent the weight control result. 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 then examine the weight setting of w_x from 0 to 1 with a granularity 0.01. The remaining weights are equally distributed to all the other parameters. For instance, if w_n is the primary factor with $w_n = 0.8$, then $\overrightarrow{w} = [0.8, 0.05, 0.05, 0.05, 0.05]$. We then 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 optimized approach: We use exhaustive search to examine the best weight settings that obtains 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 just use five parameters in our scheme so we can still obtain the result within a reasonable time. In the later section, we will investigate the cost-efficiency by comparing the time consumption in different weight control approaches.

To evaluate the estimation performance using different weight control algorithms, we use the k-fold cross-validation. To be more specific, for each training data under k peer connection, we split it into eight partitions. We sequentially select one of the partitions as the validation set, and the remaining partitions as the training set while using different weight control algorithms. To further highlight the improvement on estimation performance provided by the Packet (Bitcoin-specific) Analyses Parameters, we called the above schemes with additional use of the Packet (Bitcoin-specific) Analyses Parameters as Greedy-PA, Optimized-PA. The schemes using only the Traffic Analyses Parameters are denoted as Greedy-TA, Optimized-TA. We also include three no weight control cases including Random (randomly guessing $\tilde{k}$ from a uniform distribution), Equal-TA, and Equal-PA (two schemes using equal weights). As a result shown in Fig. 5, the accuracy performance is significantly improved for PA compared to TA. To be more specific, the MSE of different weight control algorithms using PA approach are lower than 2.282. In contrast, using only Traffic Analyses Parameters has MSE performance greater than 13.214. The optimized approach in finding the best weight setting outperforms the greedy approach, so we mainly use the optimized approach for outputting the finalized weight setting in later section.

We also measure the times spent during the weight control algorithms for Greedy-PA and Optimized-PA which are 0.0474 s and 102.1649 s respectively. Even though the costs on Optimized-PA are significantly higher than the greedy approach in the weight control stage, the estimation accuracy still outperforms the greedy one. In other words, this is a trade-off between the estimation performance and the pre-processing (weight control training) cost. Suppose one wants to obtain a better estimation performance. In that case, the pre-processing cost of Optimized-PA (yielded during the training phase) based on the exhaustive search is still acceptable.

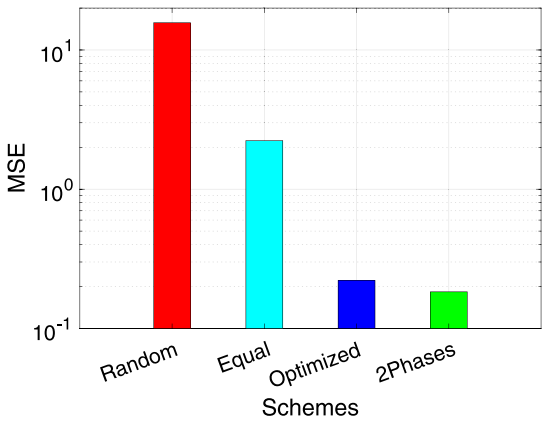
5.3. Effect of the granularity control

Since the MSE is computed by the estimation result and its real number of peer connections which should be an integer. If we further round $\widetilde{k}_x$ or finalized $\widetilde{k}$ into integers, it has a higher chance of further decreasing the error. For instance, if a testing sample with $\widetilde{k} = 4.7$ and its actual peer connection $k = 5$, then the rounding value of $\widetilde{k} = 4.7$ can make the error equal to zero. A similar situation can also be applied to $\widetilde{k}_x$ is rounded into an integer. To verify this claim, we show three different cases with the granularity control: Case 1: $\widetilde{k}_x$ and $\widetilde{k}$ are floating-point numbers; Case 2: $\widetilde{k}_x$ are still floating-point numbers but $\widetilde{k}$ are rounded into integers; Case 3: $\widetilde{k}_x$ and $\widetilde{k}$ are all integers. Note that we use the Optimized-PA to check the MSE for all cases and verify the claim. The results of MSE among different cases are 0.2483, 0.2245, and 0.1923, respectively. It thus reciprocates our claim. However, the experiments demonstrated in the following sections are based on Case 2 to provide the users some flexibility in making their own final decision.

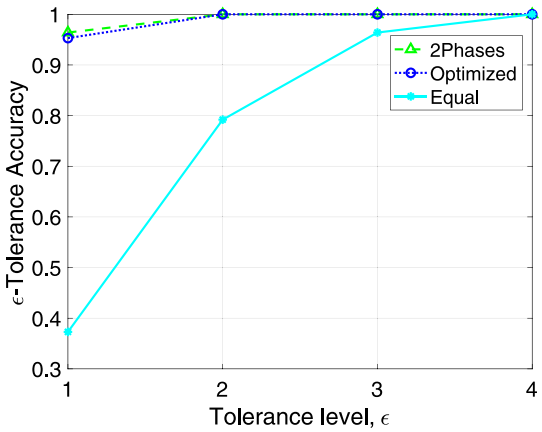
5.4. Estimation performance

5.4.1. Estimation schemes

Based on the discussion in Section 5.2, the Optimized-PA has the lowest MSE, so we take the average among the weight vectors obtained during the training phase as the finalized weight vector for the final aggregation in EA. For simplicity, we called it Optimized in the following discussion. However, based on our observation, the parameters useful for the lower peer connections might not still be beneficial for the higher peer connections cases. In view of this, we proposed a 2Phases optimized approach. As the name indicates, the 2Phases scheme has two phases: (i) the EA first classifies the networking traffic being tested between low connectivity case vs. high connectivity case and (ii) Based on the binary classification, EA uses different weight vectors obtained by the Weight Control for different cases respectively. Specifically, we separate them into the relatively lower peer connectivity case (1 to 5 peers or $1 \leq k \leq 5$) and higher peer connectivity case (6 to 10 peers or $6 \leq k \leq 10$). $k = 0$ produces zero networking traffic. The high connectivity case presents a more significant challenge because greater peer connections 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 use the finalized weight vector for the corresponding case obtained during the weight control to compute the final estimation. We also incorporate some naive approaches, i.e., Random and Equal (i.e., Equal-PA in Section 5.2), for comparison. We randomly guess k from a discrete uniform distribution within [1,10] in the Random scheme. The Equal scheme chooses equal weights for the parameters. We measure the estimation performances to all the above schemes to reveal the estimation performances using different schemes.



(a) MSE



(b) ϵ-Tolerance Accuracy

Fig. 6. Estimation performance.

5.4.2. Performance comparison in MSE and ϵ-tolerance accuracy

As a result shown in Fig. 6(a), the accuracy performance is significantly improved with a careful selection of the weight vectors conducted during the Weight Control. More importantly, the 2Phases approach in using different weight vectors for different cases (low and high connectivity) can further improve the estimation performance. The MSE of the 2Phases is only 0.1832, which significantly outperforms the MSE obtained by Random and Equal. Undoubtedly, the Random scheme makes a blind guess at k and performs the worst among all schemes. Fig. 6(b) compares the performances of the estimation schemes (we exclude the Random scheme in this comparison because of its poor MSE result) while varying ϵ from 1 to 4. All schemes increase accuracy performances when there is greater tolerance (increasing ϵ). The 2Phases scheme obtains a 96.4% 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 simply uses equal weights among the performances and therefore performs the worst accuracy performance with only 37.3% in 1-Tolerance Accuracy. We also measure the testing cost of 2Phases and Optimized because the 2Phases require additional binary classification. The testing costs of 2Phases and Optimized are 0.1057 s and 0.0825 s, respectively. Even though 2Phases requires additional effort in distinguishing the low/high peer connections of the testing sample, the cost differences between them are relatively small. Since 2Phases can further improve the estimation performance, we recommend using the 2Phases approach as the estimation scheme of our connectivity estimation engine.

5.5. Comparison with the Legacy approach

As we mentioned in the introduction, the current *Legacy* approach simply counts the network-layer identities for getting the number of peers. Therefore, it cannot detect an existence of a spoofing or Sybil attacker manipulating the peer connection based on false identifiers/IP. The comparison of our connectivity estimation engine and the *Legacy* approach thus focuses on the scenario under different attacks manipulating the peer connection (conducted in Section 6). In addition, under the normal traffic case, the actual number of peer connections is equal to the number outputted by the Legacy approach, i.e., $k = \widetilde{k}_{Legacy}$. It thus makes the MSE equaled to zero. Therefore, we did not include the result in Fig. 6 which is under the normal case. However, the provided networking traffic can be affected by the networking environment of the peers. Our connectivity estimation based on the traffic can capture this. Specifically, if there is any inconsistency between $\widetilde{k}_{Legacy}$ and $\widetilde{k}$ where $\widetilde{k} \in [0, M]$, it implies that some of peers under a bad network environment. The user of the Bitcoin node could take action such as changing peers to obtain normal traffic and get the up-to-date block as soon as possible.

We also conduct experiments to analyze the impact on the mining rate to see whether the CPU utilization of the connectivity estimation engine during testing can affect the node's mining rate. The results we obtained for using our engine and using the Legacy approach are $4.963 \cdot 10^5$ and $4.965 \cdot 10^5$ hashes per second, respectively. In fact, the mining rate without using any estimation scheme is $4.974 \cdot 10^5$ hashes per second. In other words, the mining rate reduction is limited to only 0.43% in using our engine. It reveals the lightweight property of our estimation engine.

6. 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 three attacks: the man-in-the-middle attack (which is related to the recent networking threats in blockchain such as Eclipse attack [10], routing attack [17], and Erebus attack [18]. 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 control all the traffic seen by the victim). We also prototype a PING DoS attack and a spam transaction flooding attack for verifying the effectiveness of OD.

6.1. Prototyping

6.1.1. Man-in-the-middle attack

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 $\widetilde{k}$ 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 actual 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.

6.1.2. Bitcoin PING DoS attack

We prototype the PING DoS Attack using the Bitcoin PING messages. The attacker A establishes 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 bandwidth, 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.

6.1.3. Spam transactions flooding attack

We also prototype a spam transactions flooding attack, a particular case of mempool flooding attack [19]. We consider the case that an attacker A establishes one connection to the victim X. After the connection is established, the attacker keeps sending spam transactions with a minimum relay fee to the victim, trying to flood the victim's mempool. Since the transactions are legal, it will pass the ban score mechanism in examining invalid transactions. X can still receive the Bitcoin traffic from the Mainnet through other healthy peer connections. Similar to the PING DoS attack, this attack also causes slower healthy traffic received from healthy peers.

6.2. OD effectiveness

In this prototyping, we collect 200 samples with $T_{test} = 20$ min for each attack mentioned earlier. To examine whether our OD schemes will trigger false alerts on the normal traffic data, we also collect 200 samples with $T_{test} = 20$ min under the normal networking (called Normal case in the following discussion). To show the robustness of our connectivity estimation engine, we compare the detection performance obtained by the *Legacy* approach (based on counting the network-layer identities) with the results obtained by our connectivity estimation engine facilitated with OD schemes. We evaluate the detection performance using the confusion matrix, i.e., true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Based on them, we use sensitivity (also called recall or true positive rate) as the primary metric for the testing case under attacks and specificity (also called true negative rate) as the metric for the testing case without attacks. This is because all the testing cases under attack only have TP and FN, and TP and FN constitute sensitivity. In contrast, the testing cases without attacks yield only TN and FP, and specificity is composed of TN and FP. The detection performance is summarized in Table 5.

Under the Normal case, all the schemes reach 100% specificity. It implies there are no false alerts during the testing. For the under-attack cases, the sensitivity values are all zeros using the Legacy approach because the current approach did not facilitate any defense mechanism in countering those attacks. Both OD schemes perform well in detecting those prototyped attacks. For countering the man-in-the-middle attack, both OD schemes reach 100% sensitivity because of a significant difference between the training reference and testing data. For the PING DoS attack and spam transaction flooding attack, the victim can still receive traffic from healthy peer connections. Therefore, the difference is mainly affected by the DoS attack's impact on the victim's resources. Since a Bitcoin transaction has a bigger size compared to a Bitcoin PING message, it wastes more resources on the victim and makes a more significant difference between training reference and testing data. Therefore, it is easier to detect spam transactions flooding attack than a PING DoS attack. The result shows that both OD schemes can still reach 100% specificity. However, for the PING DoS attack, the impact is not as harmful as transaction flooding, so it is relatively hard to detect. The result shows that we still obtain 98.5% specificity and 99.5% specificity for the 1.5 interquartile range rule and LoOP-Weighted, respectively. The performance of LoOP-Weighted is slightly better than the 1.5 interquartile range rule because of a more comprehensive operation of LoOP in quantifying the outlier probability based on the nearest neighbor approach. In summary, the result reveals the effectiveness of our OD schemes to counter those threats and set the $\tilde{k}$ to -1. In contrast, the Legacy approach will blindly trust all peers and estimate the number of healthy peer entities by simply counting the network-layer identities.

Table 5
OD Effectiveness (The metric of the column for the Normal case is specificity; The metric of the columns under three different attacks is sensitivity; MitM represents man-in-the-middle; txs represents transactions).

		Schemes		
		Legacy	1.5 interquartile	LoOP-Weighted
Testing cases	Normal	100%	100%	100%
	MitM attack	0%	100%	100%
	PING DoS attack	0%	98.5%	99.5%
	spam txs flooding	0%	100%	100%

6.3. Cost analysis of outlier detection

We also measure the testing cost of the OD schemes using different approaches (1.5 interquartile range rule and LoOP-Weighted) in computing $\widetilde{o}_x$ and aggregate the finalized $\widetilde{o}$. In the LoOP-Weighted approach, we use equal weights in this work. More sophisticated analysis and setting are left for future research direction. The testing overheads of the 1.5 interquartile range rule and LoOP-Weighted are 2.17 ms and 224.15 ms, respectively. The cost difference is mainly because of a simple comparison-based approach using the 1.5 interquartile range rule, where the thresholds for detecting outliers can be pre-computed. In contrast, the LoOP-Weighted approach requires a more complicated real-time computation for outputting the $\widetilde{o}_x$ and thus has more computational overhead during testing. In summary, we recommend using the 1.5 interquartile range rule due to its cost-efficiency. However, LoOP-Weighted could be more beneficial for detecting other potential threats because it provides the flexibility for the weight-setting on the outlier score.

6.4. More discussion about other threats

Even if malicious peers can collude or an attacker controls the peers to dissimulate their malicious behavior by simulating the pattern and passing the detection, what malicious peers can do to harm a Bitcoin node is craft the invalid data. However, the ban score mechanism used in the current Bitcoin core application can detect such a threat if there is an oversize data payload/or an invalid block. Our OD evaluation focuses on networking failures caused by the security threats, including the man-in-the-middle attack, PING DoS attack, and spam transactions flooding attack. While we can provide more results, we prioritize the rest of the estimation algorithm design and evaluations because they have greater research contributions. Our OD design and implementation are standard in statistical processing, although we control the detection parameters to apply to securing the Bitcoin networking estimations.

7. Related work

Bitcoin and other cryptocurrencies rely on a distributed P2P network and its connectivity. In this section, we review the literature about networking and the P2P connectivity for cryptocurrencies related to our research.

7.1. Reliability in P2P network

For dynamic distributed network systems such as P2P network, the connectivity reliability issue is of paramount importance as a significant performance metric for the network system. In [20], the authors proposed a highly reliable P2P system to ensure the effectiveness and efficiency of resource sharing in the P2P network. In [21], Xiong and Liu presented PeerTrust, a reputation-based trust supporting framework, to estimate peers' trustworthiness. They also explored mechanisms to make the PeerTrust model more robust against malicious behaviors in P2P online communities. Unfortunately, only a few literature addresses the connectivity reliability in the context of permissionless blockchain networks. Hao et al. [22] presented a trust-enhanced blockchain P2P topology that accelerates the transmission rate and retains transmission reliability. They clearly mentioned that the transmission reliability specifically refers to network trust connection. However, they focus on the reliability issue from the network-view, whereas our work focuses on the host-view. The permissionless property can make dynamic behavior even further and decrease the network system's reliability. Thus, we propose our connectivity estimation engine to address the reliability concern in permissionless Bitcoin cryptocurrency networks explicitly.

7.2. Dynamics in peer connectivity

In cryptocurrency P2P networking, *Churn* corresponds to the continuous arrival, departure, and failure of processes on peer-to-peer network and is often used to describe the dynamic peer connectivity. In [23], Stutzbach and Rejaie work on the churn models by characterizing different aspects of peer dynamics. Imtiaz et al. [24] study the intermittent network connectivity in the Bitcoin network and shows that it significantly affects the failure on compact block propagation. In addition, they perform the characterization of churn and the statistical fitting of the distributions of the lengths of up and down session. From a different point of view, Wang et al. [25] evaluate the quality of Bitcoin networks by several key metrics mainly focused on transactions and blocks. Decker et al. [26] mention that when the connectivity increases, the information propagation on the Bitcoin network can be improved. However, their research focuses on measuring the block propagation delay. In our approach, we focus on measuring the peer connectivity of a Bitcoin node at the time rather than the dynamic changes in the networking state.

7.3. Peer selection strategies

Prior literature related to the peer selection aim for optimizing the propagation delay in Bitcoin Network. Previous research control and analyze the optimal number of the outgoing connections, including based on in-computer simulations [27] and Bitcoin node implementation [28]. In [29], Fadhil et al. propose a Bitcoin Clustering Based Ping time protocol, evaluating the proximity of connectivity based on ping time latency. By using the protocol, the peers are self-cluster based on proximity. By the clustering approach, the bitcoin network can effectively decrease the transaction propagation delay.

7.4. Traffic measurement and Formal analysis

Some literature also focus on measure the traffic but have different purposes compared to our work. The healthy peer connection of a node that are responsible for information propagation is of paramount importance in the P2P network. However, most of work focuses on the topology or uncover the nodes hidden in the ecosystem. Wang and Pustogarov [30] analyze unreachable peers who do not allow inbound connections (i.e., Private node). Based on the data collected by their deploying node, they found that 86.8% of the collected IPs are unreachable. In [31], Kim et al. propose a Nodefinder tool for scanning and monitoring Ethereum's P2P network. The tool uncovers a cluttered network containing a large amount of nodes running various non-Ethereum services. Franzoni et al. [32] propose an algorithm called AToM (Active Topology Monitor) to obtain a continuously up-to-date state of the topology. The authors review network-level attacks and argue that the benefits of an open topology potentially outweigh its risks. Formal analysis of Blockchain system in asynchronous networks have also received significant attention from academic communities. In [33], Zhao et al. derive a neat bound of mining latencies to ensure the consistency property of the Nakamoto protocol. In contrast to prior literature focus on the entire cryptocurrency network, our work focus on the healthy assess of peer connection on single node for permissionless Bitcoin network.

7.5. Network-based attacks

Building on Sybil (described in Section 2), Heilman et al. [10] propose an Eclipse attack 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 [34] or double-spending attack [10]. Tran et al. propose an Erebus attack [18] to conduct a persistent eclipse attack using a malicious autonomous system (AS), creating large numbers of peer identities. In [35], Tran et al. conduct follow-up research regarding a more fundamental solution called routing-aware peering (or RAP) as a promising countermeasure against the Erebus attack. Unfortunately, the result shows that no practical RAP implementations for Bitcoin can prevent the Erebus attacks. As discussed in Section 2, our work is robust against 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). Alangot et al. [36] proposed two lightweight protocols to detect the Eclipse attack. One protocol monitors the suspicious block timestamps. The other gossip-based protocol using the natural connections of a node to the Internet to gossip about their blockchain views with contacted servers and other clients. The threat model in [36] is different from the threat we targeted in Section 6. In our prototyping, we focus on the worst case scenario that a man-in-the middle attacker controls (eclipses) all connections between the victim and the Bitcoin Mainnet and does not relay any packet to the victim.

There are some other network-based attacks in the bitcoin network, such as routing attack [17], partitioning attack [18], mempool flooding [19], DDoS attack by spam transactions [37], Bitcoin Message-based DoS (BM-DoS) attack, and Defamation attack [38]. Those attacks enabled by changing the normal traffic behavior, such as increasing the per-message count rate, can possibly be detected by our connectivity estimation engine. The verification for detecting such attacks is left for future work. Our work mainly focuses on the reliability issue raised by permissionless Bitcoin network by accurately estimating the healthy peer connections and filtering out abnormal (malicious) traffic.

8. Conclusion and future work

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. While our connectivity estimation engine is generally applicable to P2P networking (with fine-tuning

of the parameters), our work focuses on permissionless Bitcoin cryptocurrency network because of the aforementioned challenge in using identity-based connectivity estimation. Based on our implementations, we recommend the 2Phases scheme to optimize the accuracy performance as it achieves 96.4% estimation accuracy with a tolerance of one peer connection and has the MSE of 0.1832. 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.

Future research will be directed toward a more challenging public Bitcoin node case where each node can have up to 125 connections using the default setting. The case has more significant dynamic changes in peer connectivity. Therefore, more comprehensive analysis with additional beneficial parameters is required for obtaining good estimation accuracy. Another interesting research direction would be analyzing the traffic on each connection individually to see whether it is a good peer. Monitoring dynamical changes in estimation parameters in time series could be essential for the analysis.

CRediT authorship contribution statement

Hsiang-Jen Hong: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Wenjun Fan:** Methodology, Funding acquisition. **Simeon Wuthier:** Investigation. **Jinoh Kim:** Methodology, Supervision. **C. Edward Chow:** Supervision, Funding acquisition. **Xiaobo Zhou:** Supervision, Funding acquisition. **Sang-Yoon Chang:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research was supported by Colorado State Bill 18-086 and by the National Science Foundation under Grant No. 1922410. This work was also supported in part by XJTLU Research Development Funding RDF-21-02-012 and XJTLU Teaching Development Funding TDF21/22-R24-177. This paper is an extended version of the short paper published at IEEE/ACM International Symposium on Quality of Service (IWQoS), 2021 [39]. The authors extend the previous work by using a more standard approach for the outlier detection component, including the 1.5 interquartile range rule and the Local Outlier Probabilities (LoOP). Additionally, we conduct preliminary analyses for the parameters we monitor for our estimation, which better establishes the control parameters' selection.

References

[1] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, W. Knottenbelt, XCLAIM: Trustless, interoperable, cryptocurrency-backed assets, in: 2019 IEEE Symposium on Security and Privacy (S&P), 2019, pp. 193–210.

[2] H. Tian, K. Xue, X. Luo, S. Li, J. Xu, J. Liu, J. Zhao, D.S.L. Wei, Enabling cross-chain transactions: A decentralized cryptocurrency exchange protocol, IEEE Trans. Inf. Forensics Secur. 16 (2021) 3928–3941.

[3] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2009, Cryptography Mailing List at https://Metzdowd.Com.

[4] G. Wood, Ethereum: A secure decentralised generalised transaction ledger, 2014, URL http://gavwood.com/paper.pdf.

[5] F. Ritz, A. Zugenmaier, The impact of uncle rewards on selfish mining in ethereum, in: Proceedings of the 2018 IEEE EuroS&P Workshops, 2018, pp. 50–57, http://dx.doi.org/10.1109/EuroSPW.2018.00013.

[6] S.-Y. Chang, Y. Park, S. Wuthier, 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.

[7] J. Kim, M. Nakashima, W. Fan, S. Wuthier, X. Zhou, I. Kim, S.-Y. Chang, Anomaly detection based on traffic monitoring for secure blockchain networking, in: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2021, pp. 1–9, http://dx.doi.org/10.1109/ICBC51069.2021.9461119.

[8] J. Kim, M. Nakashima, W. Fan, S. Wuthier, X. Zhou, I. Kim, S.-Y. Chang, A machine learning approach to anomaly detection based on traffic monitoring for secure blockchain networking, IEEE Trans. Netw. Serv. Manag. (2022) 1, http://dx.doi.org/10.1109/TNSM.2022.3173598.

[9] G.O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, S. Čapkun, Misbehavior in bitcoin: A study of double-spending and accountability, ACM Trans. Inf. Syst. Secur. 18 (1) (2015).

[10] E. Heilman, A. Kendler, A. Zohar, S. Goldberg, Eclipse attacks on bitcoin's peer-to-peer network, in: Proceedings of 24th USENIX Security Symposium, 2015, pp. 129–144.

[11] P. Rousseeuw, M. Hubert, Robust statistics for outlier detection, Wiley Interdisc. Rew.: Data Min. Knowl. Discov. 1 (2011) 73–79, http://dx.doi.org/10.1002/widm.2.

[12] H.-P. Kriegel, P. Kröger, E. Schubert, A. Zimek, LoOP: Local outlier probabilities, in: Proceedings of the 18th ACM CIKM, New York, NY, USA, 2009, pp. 1649–1652.

[13] H.W. Lilliefors, On the Kolmogorov-Smirnov test for normality with mean and variance unknown, J. Amer. Statist. Assoc. 62 (318) (1967) 399–402.

[14] D. dos Reis, P. Flach, S. Matwin, G. Batista, Fast unsupervised online drift detection using incremental Kolmogorov-Smirnov test, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1545–1554.

[15] Z. Li, H. Zhang, M. Masum, H. Shahriar, H. Haddad, Cyber fraud prediction with supervised machine learning techniques, in: Proceedings of the 2020 ACM Southeast Conference, 2020, pp. 176–180.

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

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

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

[19] M. Saad, L. Njilla, C. Kamhoua, J. Kim, D. Nyang, 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, 2019, pp. 285–292.

[20] G. Wepiwe, P.L. Simeonov, A concentric multi-ring overlay for highly reliable P2P networks, in: Fourth IEEE International Symposium on Network Computing and Applications, 2005, pp. 83–90.

[21] L. Xiong, L. Liu, PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities, IEEE Trans. Knowl. Data Eng. 16 (7) (2004) 843–857.

[22] W. Hao, J. Zeng, X. Dai, J. Xiao, Q. Hua, H. Chen, K. Li, H. Jin, Towards a trust-enhanced blockchain P2P topology for enabling fast and reliable broadcast, IEEE Trans. Netw. Serv. Manag. 17 (2) (2020) 904–917.

[23] D. Stutzbach, R. Rejaie, Understanding churn in peer-to-peer networks, in: Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC, 2006, pp. 189–202.

[24] M.A. Imtiaz, D. Starobinski, A. Trachtenberg, N. Younis, Churn in the bitcoin network, IEEE Trans. Netw. Serv. Manag. 18 (2) (2021) 1598–1615.

[25] B. Wang, S. Chen, L. Yao, B. Liu, X. Xu, L. Zhu, A simulation approach for studying behavior and quality of blockchain networks, in: Proceedings of the 1st IEEE International Conference on Blockchain, 2018, pp. 18–31.

[26] C. Decker, R. Wattenhofe, Information propagation in the bitcoin network, in: Proceedings of the 2013 IEEE P2P, 2013, pp. 1–10.

[27] A. Sudhan, M.J. Nene, Peer selection techniques for enhanced transaction propagation in bitcoin peer-to-peer network, in: Proceedings of 2018 Second International Conference on Intelligent Computing and Control Systems, 2018, pp. 679–684.

[28] S. Wuthier, P. Chandramouli, X. Zhou, S.-Y. Chang, Greedy networking in cryptocurrency blockchain, in: W. Meng, S. Fischer-Hübner, C.D. Jensen (Eds.), ICT Systems Security and Privacy Protection, Springer International Publishing, Cham, 2022, pp. 343–359.

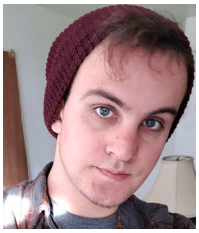
[29] M. Fadhil, G. Owenson, M. Adda, Locality based approach to improve propagation delay on the bitcoin peer-to-peer network, in: Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 556–559.

[30] L. Wang, I. Pustogarov, Towards better understanding of bitcoin unreachable peers, in: CoRR, 2017, URL https://arxiv.org/abs/1709.06837.

[31] S.K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, M. Bailey, Measuring ethereum network peers, in: Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC, 2018, pp. 91–104.
[32] F. Franzoni, X. Salleras, V. Daza, Atom: Active topology monitoring for the bitcoin peer-to-peer network, Peer-To-Peer Netw. Appl. 15 (2022) 408–425.
[33] J. Zhao, J. Tang, Z. Li, H. Wang, K.-Y. Lam, K. Xue, An analysis of blockchain consistency in asynchronous networks: Deriving a neat bound, in: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), 2020, pp. 179–189.
[34] K. Nayak, S. Kumar, A. Miller, E. Shi, Stubborn mining: Generalizing selfish mining and combining with an eclipse attack, in: Proceedings of the 2016 IEEE EuroS&P, 2016, pp. 305–320, http://dx.doi.org/10.1109/EuroSP.2016.32.
[35] M. Tran, A. Shenoi, M.S. Kang, On the routing-aware peering against network-eclipse attacks in bitcoin, in: 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021, USENIX Association, 2021, pp. 1253–1270.
[36] B. Alangot, D. Reijsbergen, S. Venugopalan, P. Szalachowski, K.S. Yeo, Decentralized and lightweight approach to detect eclipse attacks on proof of work blockchains, IEEE Trans. Netw. Serv. Manag. 18 (2) (2021) 1659–1672, http://dx.doi.org/10.1109/TNSM.2021.3069502.
[37] J. Zhang, Y. Cheng, X. Deng, B. Wang, J. Xie, Y. Yang, 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, 2020, pp. 1–6.
[38] W. Fan, S. Wuthier, H.-J. Hong, X. Zhou, Y. Bai, S.-Y. Chang, The security investigation of ban score and misbehavior tracking in bitcoin network, in: 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), 2022, pp. 191–201.
[39] H.-J. Hong, W. Fan, S. Wuthier, J. Kim, X. Zhou, C.E. Chow, S.-Y. Chang, Robust P2P connectivity estimation for permissionless bitcoin network, in: 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), 2021, pp. 1–6.

Hsiang-Jen Hong received the Ph.D. degree from the Department of Computer Science and Information Engineering at National Taiwan University of Science and Technology, Taiwan, in 2018. He is currently a post-doctoral research associate at the University of Colorado, Colorado Springs. His research interests are computer networking, blockchain, cybersecurity, and combinatorial optimization.

Wenjun Fan is currently an assistant professor with Xi'an Jiaotong-Liverpool University (XJTLU). Before joining XJTLU, he worked with University of Kent, Canterbury as postdoc 2017–2019, and University of Colorado, Colorado Springs as research associate 2019–2021, respectively. He received the Ph.D. degree in telematics systems engineering from Technical University of Madrid (UPM) in 2017. His research interests include cybersecurity, cloud computing, network softwarization, blockchain and machine learning.

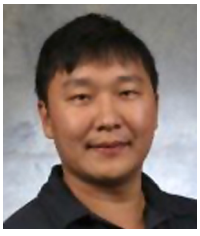


Simeon Wuthier is a Ph.D. student from the Department of Computer Science at the University of Colorado, Colorado Springs. His research is in theoretical computer science, cryptography, and distributed ledger technology.

Jinoh Kim received his Ph.D. degree in Computer Science from University of Minnesota, Twin Cities. He is currently an Associate Professor of Computer Science at Texas A&M University-Commerce and an Affiliate Faculty Scientist at Lawrence Berkeley National Laboratory. His main research interest lies in the area of networked/distributed systems with the focuses on performance, reliability, scalability, visibility, and security, with data-driven analytics and machine intelligence.

C. Edward Chow is Professor of Computer Science at the University of Colorado Colorado Springs. He got his Ph.D. in Computer Science Degree from University of Texas at Austin 1985. He served as a Member of Technical Staff with Bell Communications Research 1986–1991. His research is focused on the improvement of the security, reliability and performance of network systems.

Xiaobo Zhou obtained the BS, MS, and Ph.D. degrees in Computer Science from Nanjing University, in 1994, 1997, and 2000, respectively. Currently he is a professor of the Department of Computer Science, University of Colorado, Colorado Springs. His research lies in Cloud computing and datacenters, BigData parallel and distributed processing, autonomic and sustainable computing. He was a recipient of the NSF CAREER Award in 2009.



Sang-Yoon Chang received the B.S. and Ph.D. degrees from the Department of Electrical and Computer Engineering at University of Illinois at Urbana–Champaign in 2007 and 2013, respectively. He is an associate professor at the Computer Science Department at University of Colorado Colorado Springs. He was a postdoctoral fellow with the Advanced Digital Sciences Center. His research is in security, networking, wireless, cyber–physical systems, and applied cryptography.