RESEARCH ARTICLE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
www.rspsciencehub.com
Vol. 07, Issue 08 August

http://dx.doi.org/10.47392/IRJASH.2025.082

An Efficient Bitcoin Network Topology Discovery Algorithm for Dynamic Display

Shruti Shetti[1], Sinchana Dsa[2], Manjunath Mudda[3]
[1,2]PG – Master of Computer Applications, Dayananda Sagar College of Engineering, Bangalore, Karnataka, India.
[3]Assistant Professor, Master of Computer Applications, Dayananda Sagar College of Engineering, Bangalore, Karnataka, India.
Emails: *shrutishetti089@gmail.com[1], sinchanadsa18@gmail.com[2], Manjunath-mcavtu@dayanandasagar.edu[3]*

Article history
Received: 12 July 2025
Accepted: 28 July 2025
Published: 26 August 2025

Keywords:
Block chain, Network topology discovery Bitcoin Stability coefficient Dynamic updates

Abstract
The Bitcoin network is made up of a huge number of nodes, which makes it difficult and time-consuming for users to figure out how everything is connected. To make this easier, we've come up with a new algorithm that uses lightweight "probe" nodes to quickly gather information about the network. On top of that, we've built a clustering system that groups stable nodes together, so we can map out the network faster and more efficiently. We've also created a flexible, real-time visualization tool that shows how the network is structured in layers. Our tests show that this approach cuts down communication overhead by about 72%, while still maintaining 95% accuracy in mapping the network. Even better, the same method can be used for other blockchain networks with similar setups.

1. Introduction

Over the years, blockchain has been used in many different areas. But even with its potential, the technology isn't perfect. It still has some weaknesses, especially in how different parts of the system agree on transactions — known as the consensus layer. These flaws open the door to various attacks and scams, such as double-spending, Ponzi schemes, fake smart contracts, money laundering, and online gambling. Researchers have been actively working on ways to detect and prevent these threats:

- One team developed a method to spot Ponzi schemes on Ethereum by studying transaction patterns.
- Another group used behaviour graphs to detect suspicious smart contracts.
- Some explored how attackers hide phishing attempts and tested how well detection tools work in those cases.
- Others studied "Pump and Dump" scams and created machine learning models to catch them.
- There's even work on identifying the people behind anonymous Bitcoin transactions by analysing transaction sizes and timing.

Beyond these issues, the network layer of blockchain — the part that handles communication between nodes — also faces serious threats. These include:

- DDoS attacks that flood the network, Eclipse attacks that isolate a node from the rest of the network, and Network partitioning, where the network is split apart.

- Researchers have proposed various strategies to protect against these attacks. For example:
- Some have categorized different attack types and suggested how to defend against them. [1]
- Others have shown how hackers can hijack internet traffic to isolate Bitcoin nodes.
- A few have even demonstrated how attackers could target multiple blockchain platforms at once.
- There's also ongoing work to make transactions more anonymous at the network level by hiding where messages originate.

To fight off these types of attacks, it's important to understand how the blockchain network is structured — its topology. But here's the problem: current tools for discovering network structure are slow, expensive, and often outdated. [2]

What We Did:

To solve this, our work focuses on improving how we explore and visualize blockchain networks. Here's what we contributed:

- Faster, Smarter Discovery: We made existing discovery tools more efficient by focusing on the most reliable nodes. This makes the process faster and lighter.
- Layered Mapping: We designed a clustering system that organizes nodes into layers. This helps us explore the network structure in parallel — like mapping different zones at once.
- Real-Time Visualization: We created a tool to show how the network is structured in real time. It helps spot unusual behavior or attacks quickly.

Paper Overview:

- **Section 2** looks at what other researchers have done in this area.
- **Section 3** outlines background work.
- **Section 4** explains our algorithm.
- **Section 5** presents our test results.
- **Section 6** wraps things up and shares final thoughts.

2. Related Work

2.1. Network Topology Discovery

Understanding how a network is structured — also called network topology discovery — involves using tools like ICMP, ARP, and SNMP. These tools help check which devices are active on a network and collect information about them. There are different automated ways to do this. Some use SNMP, others rely on standard communication protocols, and some work through routing protocols. Beyond just mapping the network, researchers have also worked on making these networks more stable. For example, Chen used genetic algorithms to improve how the network is designed, making it stronger against potential attacks. Yan et al. focused on keeping the network stable, even when some devices or nodes aren't reliable. [3]

2.2. Bitcoin Network Topology Discovery

The use of network discovery techniques in Bitcoin research started back in 2014. One of the first major contributions who showed that it was possible to link Bitcoin users' pseudonyms to their real IP addresses — even if they were using NAT or hidden behind ISP firewalls. This proved that deanonymizing Bitcoin users was possible in some cases. Following that, Grundmann explored how to group different Bitcoin addresses that belong to the same node. This helped them estimate how many nodes in the network could actually be reached. They also refined their results by factoring in Bitcoin's limit on the number of active connections a node can have. It took things a step further by designing a protocol to monitor how reachable nodes in the Bitcoin network are connected. Their research revealed that up to 20% of nodes could be potentially malicious. It built a tool called a Bitcoin network sniffer, which was able to detect 9,515 active nodes in just 1.5 hours. They also introduced a method to figure out how these nodes were connected, giving a clearer picture of the network's structure. Later, it suggested a time-based method to estimate node connections in the network. Even though it was tested in a simulated environment, it managed to achieve around 40% accuracy and recall in identifying links between nodes. Lastly, it created a framework called Node Maps to study blockchain networks. Their tool examined where nodes are located around the world, which hosting services they use, and what software they run. Their findings showed that Bitcoin has the most widespread global presence among blockchain networks. Figure 1 shows Initial Handshake Between Peers

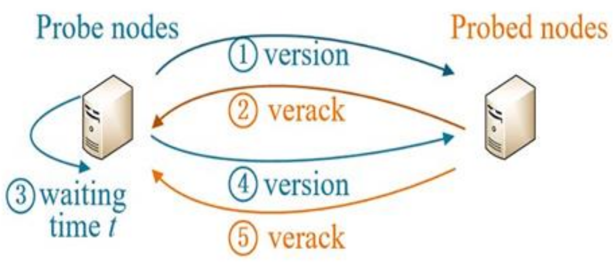

Figure 1 **Initial Handshake Between Peers**

3. Preliminaries

3.1. Bitcoin Network Nodes Establishing Connections

Most of the time, nodes use TCP port 8333 to make these connections, which is the standard port for Bitcoin traffic. However, other ports can also be used if needed. Once a connection is made, the new node starts a handshake process by sending a version message that includes some basic information for identification and compatibility. (This whole process is shown in Figure 1, and the details of the version message are listed in Table 1). There are two main ways for a new node to find other peers:

Using DNS Seeds: These are special DNS servers that give the node a list of known Bitcoin node IP addresses.

Here we can see the table for the 2 types

- Version message table 1
- Addr message table 2

Table 1 **Version Message**

Property	Meaning
nVersion	Defines the version of the Bitcoin P2P protocol "spoken" by the client (e.g., 70002).
nLocalServices	A list of local services supported by the node. Currently, only NODE_NETWORK is supported.
nTime	Current time
addrYou	IP address of the remote node visible to the current node.
addrMe	Local IP address found by the local node.
subver	Subversion number indicating the type of software currently running on the node (e.g., "/Satoshi:0.9.2.1/").
BaseHeight	Block height of the current node blockchain.

addresses of Bitcoin listening nodes. Some DNS seeds are custom implementations of Berkeley Internet Name Daemon (BIND) that return a random subset of the list of Bitcoin node addresses collected from searchers or long-running Bitcoin nodes. Alternatively, the IP address of a Bitcoin node can be manually specified as the Bitcoin seed node via the seed node command. Once a connection is created using the initial seed node, it is possible to disconnect and perform a connection probe via the newly discovered peer. Figure 2 shows Address Propagation and Discovery

Table 2 **Addr Message**

Property	Meaning
IP address count	Number of IP addresses, up to 1000.
Time	The services announced by the node in its version message.
Services	IP address of the remote node visible to the current node.
IP address	IPv6 addresses in large byte order; IPv4 addresses can be provided as IPv6 addresses for IPv4 mapping.
Port	Port numbers are listed in large alphabetical order.

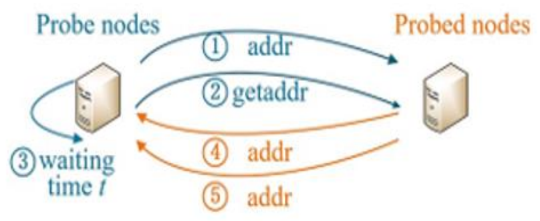

Figure 2 **Address Propagation and Discovery**

3.2. Bitcoin Node Synchronization

Once a new Bitcoin node connects to a few peers, it shares its own IP address by sending out an addr message to those peers. (The details of this message are shown in Table 2, and the whole process is illustrated in Figure 2.) After receiving this message, the neighbouring nodes pass it along to their own peers. This helps spread the new node's information across the network and improves its chances of maintaining stable connections. Alternatively, the new node can send out a getaddr message to ask its peers for a list of other nodes they know. This allows it to quickly discover more connections and share its own presence so others can find it too. To stay connected and be a reliable part of the network, each node needs to build links with multiple peers. This creates multiple paths for communication in case some connections drop. Since Bitcoin nodes can go online or offline at any time, the network has to deal with a lot of changes and uncertainty. That's why nodes are always doing two things:

- Looking for new peers if old ones go offline. [4]
- Helping new nodes find their way into the network.

Interestingly, a new node only needs one good connection to get started. That one connection can lead it to more peers, which then introduce it to even more — so there's no need to connect to too many at once. After setup, the node saves a list of peers it connected with successfully. If it ever restarts, it tries those same peers again first. And if none of them respond, the node turns to trusted seed nodes to reconnect. To make sure connections stay alive, nodes send occasional keep-alive messages. But if a connection stays completely inactive for 90 minutes, it's considered dead, and the node starts looking for a new peer. This whole process lets the Bitcoin network grow, shrink, and recover as needed — all on its own, without any central control. It's a flexible and self-sustaining system that keeps things running smoothly to other nodes when they start up. A node needs only one connection when it starts because the first node can introduce it to its peers, which in turn will provide further introductions. A node that connects to a large number of other peer nodes is both unnecessary and a waste of network resources. Once the start-up is complete, the node remembers its most recently successfully connected peer nodes, so that when restarted, it can quickly re-establish connections with the previous network of peer nodes. If the peer node of the previous network does not respond to the connection request, the node can use the seed node for restarting. If the established connection is not communicating with data, the host node will periodically send messages to maintain the connection. If a node continues a connection for up to 90 min without any communication, it is considered to have been disconnected from the network, and the network will start looking for a new peer node. Therefore, the Bit- coin network dynamically adjusts to changing nodes and network issues at any time, and it can organically scale up or down without centralized control. [5]

4. Our Work

4.1. System Architecture

To understand how the Bitcoin network is structured, we start by placing a probe node inside the network. This special node collects information about how other nodes communicate, especially at the application level. By analyzing this data, the probe can figure out which nodes are connected to each other and gradually map out the entire network. As it gathers more connection data, it can also identify which nodes are active and stable — which helps simplify and speed up the discovery process. There are about 9,000 active nodes in the Bitcoin network, and they generate close to 288,000 transactions every day. These nodes constantly update their connection details, especially since new nodes are always joining and others are leaving. Because of this, our algorithm needs to be very time-sensitive, since the network structure can change quickly and frequently. Figure 3 shows how the system is set up. In this setup, we have seed nodes — these are the starting points that help us gather connection information. Seed nodes are full Bitcoin nodes, and their IP addresses are stored on DNS servers maintained by the community. The connected nodes are those that are directly linked to these seed nodes. Each Bitcoin node can have up to 1,000 connections, and they keep in sync by regularly sharing information with their peers. We also define second-order connected nodes, which are basically the peers of connected nodes — and the structure continues in layers, with each level referred to as the n-th order. From the viewpoint of the probe node, the entire network can be seen as a series of layered connections. In the diagram (Figure 3), the probe node is the one running our algorithm. Its job is to spot which nodes are active and consistently online. In the figure, different types of lines represent different kinds of connections: a dashed line without an arrow shows a general link between nodes, a solid line with an arrow means the probe can connect directly to that node, and a dashed arrow indicates the probe needs help from another node to reach it. Figure 3 shows System Architecture of the Bitcoin Topology Discovery Algorithm

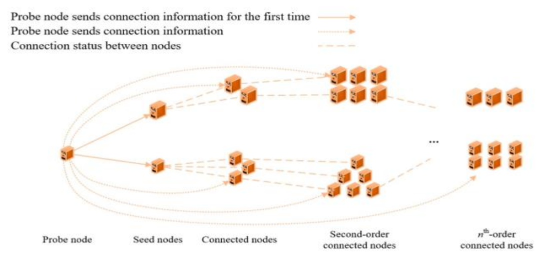
Probe node sends connection information for the first time
Probe node sends connection information
Connection status between nodes
...
Probe node
Seed nodes
Connected nodes
Second-order connected nodes
n^{th}-order connected nodes

Figure 3 System Architecture of the Bitcoin Topology Discovery Algorithm

4.2. Network Topology Discovery Process

The core idea of this lightweight algorithm is simple: we use a probe node to explore the Bitcoin network by collecting data on how different nodes are connected. It looks for active nodes that can be connected, then builds a map showing how they relate to each other. The formula for updating a node's stability score is:

$$k' = \beta \cdot kn - 1 + (1 - \beta) \cdot kn$$

Where:

- k_{n-1} is the node's stability from the last round,
- k_n is the current round's value,
- k' is the final updated score, and β is a balancing factor (set to 0.9) that gives more weight to past values to smooth out sudden changes. [6]

This strategy (explained in Algorithm 1) helps reduce the number of network requests, since the probe doesn't need to ask every node for fresh data all the time. Instead, it focuses on stable nodes whose results can be reused in future rounds. This makes the whole process faster and less demanding on the network — without sacrificing accuracy. Figure 4 shows Flow Chart of the Network Topology Discovery Algorithm [7]

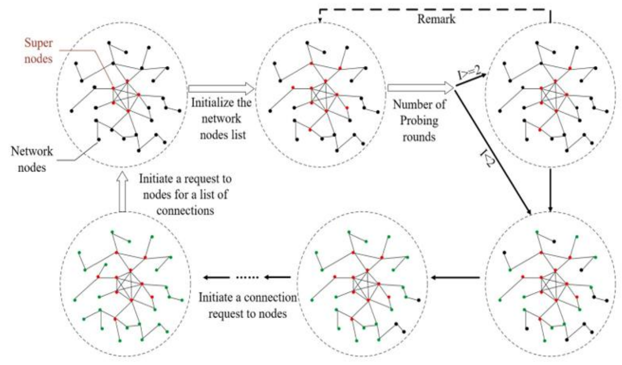
Super nodes
Network nodes
Initialize the network nodes list
Remark
Number of Probing rounds
I>=2
I<2
Initiate a request to nodes for a list of connections
Initiate a connection request to nodes

Figure 4 Flow Chart of the Network Topology Discovery Algorithm

Step 2: Identify the most stable nodes within the known network topology. To achieve this, perform an intersection of the two extracted node files — focusing solely on the nodes themselves, without considering their connection details. [8]

Step 3: Integrate the intersected node set into the connected nodes of the known network. If a node appears in both files and its connection status remains unchanged across both probing rounds, its connection information is preserved. This stable node and its connections are then added to the network topology and recorded in the result file.

Step 4: Perform a final filtering of the retained nodes and their connections. Some nodes may have been included in the earlier steps but do not maintain valid connections, and therefore, do not meet the criteria for stable nodes. These are filtered out, and the refined data is stored in the corresponding Redis database, ready for the next probing cycle. [9]

Algorithm 1: Network topology discovery algorithm.

```
Input: Blockchain initial network node list InitNodes.
Output: Blockchain network topology result.
1  $k = 0, \beta = 0.9, x = 0.5$;
2  Nodes = [InitNodes], result = [];
3  while Nodes not NULL do
4      if $k_n < k$ then
5          foreach node in Nodes do
6              Discovered=getaddr(node);
7              Nodes.append(Discovered);
8              foreach Discnode in Discovered do
9                  result.append([node, Discnode]);
10             end
11         end
12         $k'_n = \beta k_{n-1} + (1-\beta)k_n$;
13     end
14     else
15         $k_n = \ln(e-1)\left(k_{n-1} + \frac{1}{e-1}\right)$;
16     end
17 end
18 return result;
```

Definition 1 (Stable node).

The current node is $v1$, which has the following conditions:

(1) $G = <V, E>, V = \{v_1, v_2, ..., v_n\}$,
$E = \{v_1v_2, v_1v_3, v_1v_4, ..., v_1v_n\}$

(2) $M_1 = \{G_{11}, G_{12}, ..., G_{1n}\}, M_2 = \{G_{21}, G_{22}, ..., G_{2n}\}$

(3) $G \subset M_1, G \subset M_2$

Algorithm 2: Find the set of stable nodes.

```
Input: Nodes dictionary dic0, dic1
Output: Stable nodes set R
1  R = [];
2  N = []; // Initialize array
3  if $len_{file} \geq 2$ then
4      $ins_{key}$ = dict0.keys() ∩ dict1.keys();
5      foreach key in $ins_{key}$ do
6          if dict0[key]= dict1[key] then
7              N.add(key);
8          end
9      end
10     R.append(N);
11 end
```

5. Experimental Analysis

To test how well our proposed algorithm works in mapping the Bitcoin network, we built a custom probing system. This system included three probe nodes, each set up with a different approach to gather data. All of them could send connection requests to nodes within the Bitcoin network. The probe nodes were quite powerful: each ran on an Intel Xeon Silver 4214 CPU @ 2.20GHz, with 64 GB of RAM and 500 GB of storage, and used Ubuntu Linux (64-bit). The nodes were located in world. For our experiment, we used two different probing methods. [10]

- The first method was based on Bitnodes1, a public project launched by Addy Yeow in 2013. It works by broadcasting connection requests to every node in the Bitcoin network during each round of scanning.
- The second method was our own lightweight algorithm, designed to reduce network traffic by only probing the most relevant and stable nodes. [11]

The goal was to see how our new method stacks up against one of the most trusted sources for Bitcoin network data — Bitnodes. So, we used Bitnodes' server node counts as a benchmark. We started probing from an IP address beginning with 59.x.x.205, following the step-by-step process explained earlier. All three probe nodes pulled seed data from the same DNS seeder to ensure fairness. Two of them used the Bitnodes1-style method, while one used our optimized version. Once everything was set up, the nodes began by connecting to seed nodes, collecting peer lists, and using that data to form new connections. This process was repeated to explore deeper into the network. Since Bitnodes scans the network roughly every five minutes, we timed our probes to match those intervals as closely as possible — keeping any time differences within just a few seconds. We ran these probing sessions once a day over multiple days to collect enough data and ensure accuracy. We then compared the results by looking at:

- how many nodes were discovered in total,
- how many nodes matched across methods,
- how many had the same connection patterns.

On average, our method matched 94.41% of the total nodes found by Bitnodes. To improve reliability, we also ran a control experiment using Bitnodes1 from one of the probe nodes. We found that 94.72% of the nodes were the same across methods, and 95.52% of those nodes shared the same connections. [12]

6. Comparative Efficiency Analysis of Network Topology Algorithms

For this part of the experiment, we ran two probe nodes at the same time, both using seed nodes from the same DNS seeder to keep things consistent. One of the nodes used the first method (the Bitnodes1-style approach), while the other used our new lightweight algorithm. During the test, these nodes sent connection requests to around 500,000 IP addresses. We tracked and compared a few key things for both methods — like how accurately they mapped the network, how many connection requests they had to send, and how many connection responses they received. The results. What we found was impressive. Our lightweight algorithm reduced the number of connection requests by an average of 58.88% compared to the first method. Even better, it cut down the number of messages needed to get node connection info by 72.16%. That's a big improvement in efficiency. To take things further, we created a simulated Bitcoin network with 7,000 nodes and over 40,000 connections. This gave us a realistic environment to test and analyze how well our algorithm performs. it can quickly re-establish connections with the previous network of peer nodes. If the peer node of the previous network does not respond to the connection request, the node can use the seed node for restarting. If the established connection is not communicating with data, the host node will periodically send messages to maintain the connection. If a node continues a connection for up to 90 min without any communication, Figure 5 shows Graph, Figure 6 shows Number of Detections & Nodes, Figure 6 shows Number of Detections & Nodes, Figure 8 shows Nodes [13-15]

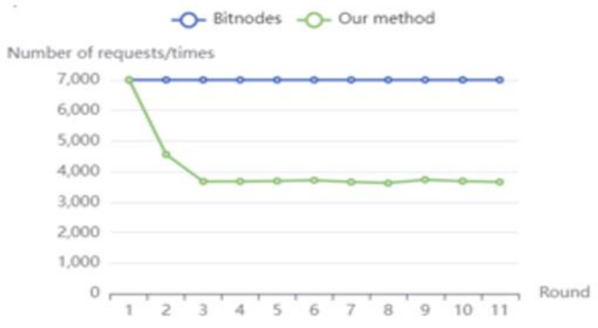

Figure 5 Graph

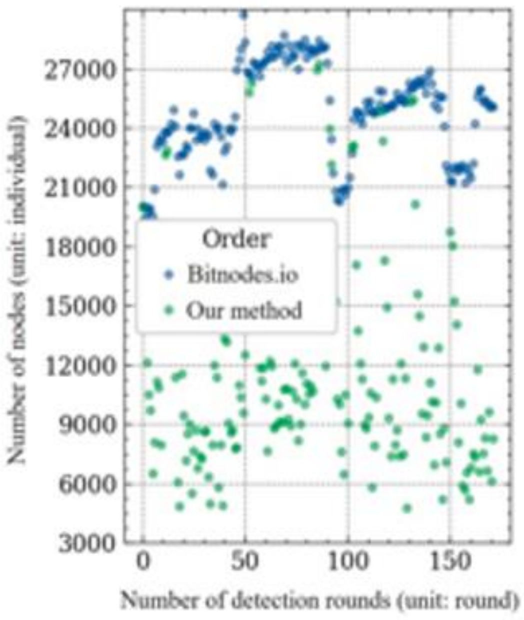

Figure 6 **Number of Detections & Nodes**

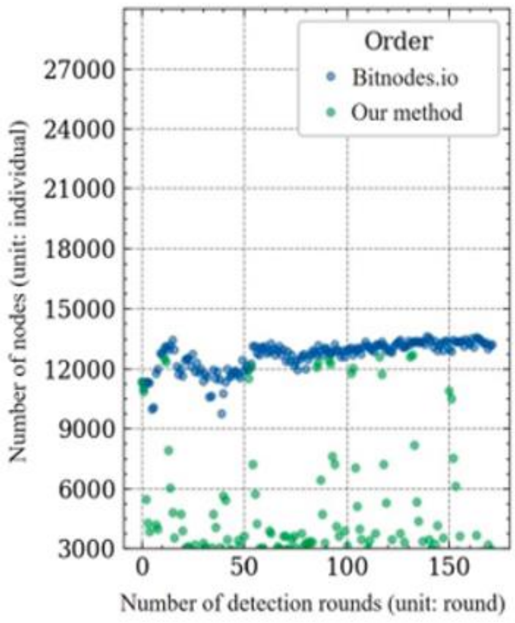

Figure 7 **Number of Detections & Nodes**

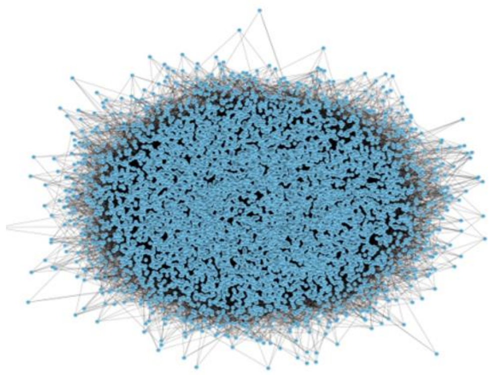

Figure 8 **Nodes**

Conclusions

We've developed a new algorithm to map out the Bitcoin network that uses lightweight scout nodes to gather information more quickly and efficiently. Compared to the commonly used Bitnodes method — which relies on scanning the entire network — our approach is noticeably faster and more efficient, even though there are small differences in the final results. What sets our method apart is that it doesn't just list active nodes — it also gathers detailed information about each node and continuously improves its discovery process. When we tested both methods side by side, our algorithm needed fewer connection requests, made fewer queries, and finished the discovery process in less time, all while keeping an average accuracy of about 95%. While spreading detection across multiple probe nodes can improve accuracy, our approach still works well when run from just one node. That said, network delays and latency can slightly affect how efficiently it performs in single-node setups. We also added a feature using METIS, which helps explore the network in parallel and visualize any unusual activity in real time — making the whole system smarter and more responsive.

References

[1]. F. Franzoni, V. Daza, Clover: "an anonymous transaction relay protocol for the Bit- coin P2P network", Peer-to-Peer Network. Appl. 15 (2022) 290–303, https://doi.org/10. 1007/s12083-021-01241-z.

[2]. S. Handoko, H.L.H.S. Warnars, "Network scanning topology based on inventory using query SNMP method", in: J. Chand Bansal, K. Deep, A.K. Nagar, et al. (Eds.), Smart Data Intelligence. Algorithms for Intelligent Systems, Springer, Singapore, 2022, pp. 295–306, https://doi.org/10.1007/978-981-19-3311-0_25.

[3]. S. Zhou, L. Cui, C. Fang, et al., "Research on network topology discovery algorithm for Internet of things based on multi-protocol", in: Proceedings of the 2018 10th International Conference on Modelling, Identification and Control (ICMIC), IEEE, 2018, pp. 1–6, https:// doi.org/ 10.1109/ ICMIC.2018.8529955.

[4]. Y. Cui, Q. Zhang, Z. Feng, et al., "Topology-aware resilient routing protocol for FANETs: an adaptive Q-learning

approach", IEEE Internet Things J. 9 (19) (2022) 18632–18649, https:// doi.org/ 10.1109/JIOT.2022.3162849.

[5]. N. Chen, T. Qiu, Z. Lu, et al., "An adaptive robustness evolution algorithm with self-competition and its 3D deployment for Internet of things", IEEE/ACM Trans. Network. 30 (1) (2022)368–381, https:// doi.org/10.1109/TNET.2021.3113916.

[6]. P. Yan, S. Choudhury, F. Al-Turjman, et al.," An energy-efficient topology control algorithm for optimizing the lifetime of wireless ad-hoc IoT networks in 5G and B5G", Computer. Commun. 159 (2020) 83–96, https:// doi.org/ 10.1016/ j.com com.2020.05.010.

[7]. A. Biryukov, D. Khovratovich, I. "Pustogarov, Deanonymisation of clients in Bitcoin P2P network", in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2014, pp. 15–29, https:// doi.org/ 10.1145/ 2660267.2660379.

[8]. M. Grundmann, M. Baumstark, H. Hartenstein, "Estimating the peer degree of reach- able peers in the Bitcoin P2P network", arXiv preprint arXiv:2108.00815, 2021, https:// doi.org/ 10.48550/ arXiv.2108 .00815.

[9]. M. Grundmann, M. Baumstark, H. Hartenstein, "On the peer degree distribution of the Bitcoin P2P network", in: Proceedings of the 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, 2022, pp. 1–5, https://doi.org/10. 1109/ ICBC 54727. 2022.9805511.

[10]. R. Li, J. Zhu, D. Xu, et al., "Bitcoin network measurement and a new approach to infer the topology", China Commun. 19 (10) (2022) 169–179, https://doi.org/10.23919/ JCC.2022.00.030.

[11]. T. Klonowski, Bitcoin network topology discovery using timing analysis, Ph.D. thesis, Technische Universität Wien, 2023.

[12]. A. Howell, T. Saber, M. Bendechache, "Measuring node decentralisation in blockchain peer to peer networks", Blockchain Res. Appl. 4 (1) (2023) 100109, https://doi.org/ 10.23919/JCC.2022.00.030.

[13]. M. Sallal, R. de Fréin, A. Malik, et al., "An empirical comparison of the security and performance characteristics of topology formation algorithms for Bitcoin networks", Array 15 (2022) 100221, https:// doi.org/ 10.1016/j.array.2022.100221

[14]. B.G. Gebraselase, B.E. Helvik, Y. Jiang, "Bitcoin P2P network measurements: a testbed study of the effect of peer selection on transaction propagation and confirmation times", IEEE Trans. Network. Serv. Manag. 19 (4) (2022) 3975–3987, https://doi.org/ 10.1109/TNSM.2022.3216955.

[15]. H.-J. Hong, W. Fan, S. Wuthier, et al., "Robust P2P networking connectivity Estima- tion engine for permissionless Bitcoin cryptocurrency", Computer. Network. 219 (2022) 109436, https:// doi.org/10.1016/j.comnet.2022.109436.