# Improving Bitcoin Transaction Propagation Efficiency through Local Clique Network

Kailun Yan[1], Jilian Zhang[2] and Yongdong Wu[2]

[1]*College of CS and Information Engineering, Guangxi Normal University, Guilin, China*
[2]*College of Cyber Security, Jinan University, Guangzhou, China*
*Email: victory_yan@foxmail.com,zhangjilian@jnu.edu.cn,wuyd007@qq.com*

**Bitcoin is a popular decentralized cryptocurrency, and the Bitcoin network is essentially an unstructured peer-to-peer (P2P) network that can synchronize distributed database of replicated ledgers through message broadcasting. In the Bitcoin network, the average clustering coefficient of nodes is very high, resulting in low message propagation efficiency. In addition, average node degree in the Bitcoin network is also considerably large, causing high message redundancy when nodes use the gossip protocol to broadcast messages. These may affect message propagation speed, hindering Bitcoin from being applied to scenarios of high transactional throughputs. To illustrate, we have collected single-hop propagation data of transactions of 366 blocks from Bitcoin Core. The analysis results show that transaction verification and network delay are two major causes of low transaction propagation efficiency. In this paper, we propose a novel P2P network structure, called *local clique network* (LCN), for message broadcasting in the Bitcoin network. Specifically, to reduce transaction validation latency and message redundancy, in LCN local nodes (logically) form cliques, and only a few nodes in a clique broadcast messages to the other cliques, instead of each node sending messages to its neighboring nodes. We have conducted extensive experiments, and the results show that message redundancy is low in LCN, and message propagation speed increases significantly. Meanwhile, LCN exhibits excellent robustness when average node degree remains high in the Bitcoin network.**

*Keywords: Bitcoin network, peer-to-peer network, transaction propagation, blockchain*

*Received 00 January 2009; revised 00 Month 2009*

## 1. INTRODUCTION

Bitcoin is one of the most popular digital cryptocurrencies [1–3], exhibiting characteristics of decentralization and anonymity. First introduced in Bitcoin, blockchain [4–7] combines techniques from cryptography, consensus mechanism, P2P network, and other related fields to realize a distributed database to store transactions [8,9]. Bitcoin network is an unstructured network, and it uses gossip protocol [10] to synchronize messages. Nodes broadcast transactions and blocks via a peer-to-peer (P2P) network. The gossip protocol is simple and exhibits good robustness and scalability, but it suffers from security risks such as flood attacks [11] and broadcast storms [12, 13]. In contrast, Ethereum [14, 15] is a structured network that employs the Kademlia algorithm [16] and distributed hash table (DHT) technology to route and locate data in a P2P network accurately. Gencer et al. pointed out that Ethereum nodes are widely distributed compared to Bitcoin nodes, but Ethereum nodes have less spare bandwidth [17]. In general, neither Bitcoin nor Ethereum has apparent advantages over the other.

Nodes have four major functionalities in the Bitcoin network, i.e., wallet, mining, database, and network routing. Each node independently verifies transactions and blocks without relying on any other nodes [18]. Researchers have conducted extensive studies to analyze the Bitcoin network [19–21]. One observation is that the spread of transactions is much slower than blocks because transactions mainly rely on wallet nodes [20]. Meanwhile, each wallet node broadcasts messages received to the other nodes, resulting in high message redundancy. The slow transaction propagation speed prevents Bitcoin from being suitable for high transaction throughput scenarios.

Currently, there exists a lot of research work on the security and privacy issues of Bitcoin [22–24], however, less attention has been paid to improving the efficiency of transaction propagation [25]. Decker

et al. [19] suggest forwarding small transactions directly to reduce the propagation delay, but this incurs a risk that invalid transactions may spread across the Bitcoin network. Erlay [26] is a low-bandwidth transaction propagation protocol, which avoids linear increase in bandwidth consumption as the number of connections increases, at the cost of larger transaction latency. Conflux [27] uses Shrec [28] to improve the propagation of effective transactions and increase blockchain throughput, however Shrec requires extra CPU costs. BCBPT [29] is a proximity-aware extension to the current Bitcoin protocol that groups Bitcoin nodes based on *ping* latencies between nodes. INDF [30] consists of two algorithms for a node to find influential nodes to broadcast transactions. Both BCBPT and INDF may have Matthew effects, aggravating the centralization of Bitcoin network.

We analyze real transaction data in the Bitcoin network in Section 3, and analysis results show that transaction verification and network delay are two significant causes of low transaction propagation efficiency in the Bitcoin network. In this paper, we propose a novel network structure called *local clique network* (LCN) to improve transaction propagation efficiency.

Specifically, in LCN, local nodes (logically) form cliques. A node adopts different transaction propagation strategies with respect to nodes in the same clique (intra-clique) and nodes in other cliques (inter-clique), in order to improve transaction propagation efficiency and reduce message redundancy. Compared to the current work, our solution does not rely on additional prior knowledge, easy to implement and can improve transaction propagation efficiency in the Bitcoin network. Moreover, our approach is compatible with the transaction propagation protocol currently used in Bitcoin network.

In this paper, we made the following contributions:

- We collect real transaction data from Bitcoin Core, and conduct extensive data analysis to show that transaction propagation delay is mainly caused by transmission delay and verification delay in the Bitcoin network.
- We propose a novel network structure called *local clique network* (LCN), which logically groups the nodes into different cliques. We design two novel propagation strategies (i.e., strategy (a) and (b)) for inter-clique and intra-clique nodes to forward messages, aiming to reduce inter-clique and intra-clique message redundancy in LCN. Also, the existing Bitcoin network can easily incorporate our propagation strategy (a).
- We have conducted extensive experiments, and the results show that LCN can effectively increase transaction propagation efficiency and reduces message redundancy. We have also found that the Bitcoin network achieves high transaction propagation efficiency and network robustness if propagation strategy (a) is employed.

We organize this paper as follows. First, we give a literature review in Section 2, then we analyze real transaction data to identify the major causes that affect transaction propagation efficiency in the Bitcoin network in Section 3. We present LCN and two efficient strategies for transaction propagation in Section 4, and conduct extensive experiments in section 5 to verify the effectiveness and robustness of LCN. Finally, we conclude this paper in Section 6.

## 2. RELATED WORK

Some researchers [31] identify nodes in the Bitcoin network by using external information such as IP addresses, geographic locations, etc. Christian et al. [19] analyzed blocks and transactions and concluded that Bitcoin network delay is the leading cause of blocks fork. They improved block propagation and suggested forwarding small transactions directly to reduce the time delay caused by transaction round trips. A single node itself suffers from the risk of flooding attacks in the Bitcoin network, and directly forwarding small transactions extends this kind of risk to the whole network. Therefore, it is necessary to verify a transaction before forwarding it.

Bitcoin uses proof-of-work (PoW) as the consensus mechanism to ensure the security of a distributed ledger. In order to obtain higher returns, mining nodes form mining pools to share computing power [32]. Specifically, miners and the mining pool server use a unique protocol to communicate. The mining pool server acts as a full Bitcoin node that employs the stratum protocol to communicate with nodes in the Bitcoin network [33, 34]. Lischke et al. [21] claimed that the Bitcoin network is a scale-free network [35, 36] at certain levels (e.g., time, businesses, and country), and mining pools have a large number of connections. Mining pools try to receive more transactions in time (meaning more transaction fees), however they are reluctant to forward transactions to prevent other mining pools from receiving these transaction fees [37, 38]. Donet and Antoni [20] confirmed this phenomenon, and they found that transaction propagation speed is much slower than blocks, i.e., in the Bitcoin network, it takes less than 22 seconds to broadcast 50% blocks to 25% of the nodes. In contrast, it takes 17 minutes to broadcast 50% of transactions to 25% of the nodes.

Existing approaches focus on improving the propagation efficiency of transactions or blocks by finding influential nodes [30, 39] or reducing the delay between nodes through clustering [29, 40]. RepuLay [39] is a reputation-based relay protocol that effectively accelerates the propagation of transactions and optimizes the usage of nodes bandwidths. BCBPT [29] groups bitcoin nodes based on the ping delay between nodes to improve broadcast efficiency. BlockP2P-EP [40] uses

the K-Means algorithm for gathering proximity peer nodes into clusters and conducts a parallel spanning-tree broadcast algorithm. MempoolSync [41] is a new synchronization protocol that sends transactions to peers in an effort to alleviate the impact of churn and keep mempools of nodes synchronized. Zhang et al. [42] derive a novel influence time minimization (ITM) problem and propose a greedy-based algorithm for fast transaction broadcasting in Bitcoin network.

In the Bitcoin network, if we exclude mining pools, casinos, exchanges, wallet nodes are primary producers of transactions, wallet nodes are mainly responsible for transaction forwarding. Therefore, we mainly focus on improving transaction propagation efficiency for wallet nodes, which form a subnetwork (i.e., ignoring nodes such as mining pools, casinos, and exchanges) of the Bitcoin network [21]. There are two kinds of wallet nodes, namely full nodes and lightweight nodes. A full node stores complete blockchain data. In order to maintain a local copy of the blockchain and the transaction pool, a full node will remain online. A Lightweight node generally runs on devices with limited resources, e.g., smartphones. A lightweight node only stores the block headers of the blockchain and the transactions of this node. Light nodes cannot independently verify the legitimacy of a transaction. They can only complete a transaction by simple payment verification (SPV), while full nodes can independently verify all transactions and new blocks. Therefore, our investigation focuses on the full nodes. In the sequel, when mentioning the Bitcoin network, we refer to the subnetwork of wallet nodes (i.e., full nodes).

The Bitcoin network is essentially a complex network with the characteristics of self-organization and self-similarity. In a complex network, the degree of a node refers to the number of neighbors connected to that node, and the average node degree refers to the average of node degrees across the entire network. We usually use the following to measure a network:

(1) Characteristic path length is also called the average shortest path. The path length between two nodes refers to the minimum number of edges that one node passes to reach another node. The average of the path lengths of all nodes is defined as the *characteristic path length* of the network.
(2) Clustering coefficient is a measure of the degree of clustering of nodes in a network. Specifically, if a node has $k$ neighbors, there may be at most $\frac{k(k-1)}{2}$ edges between neighbors. The actual number of edges died by $\frac{k(k-1)}{2}$ is the node's local clustering coefficient. The average of local clustering coefficients of all the nodes is called the *average clustering coefficient*.

There is a small-world effect in the Bitcoin network [20, 43], meaning that the Bitcoin network has a high average clustering coefficient and a small average shortest path. When the probability of reconnection between nodes is 1, a small-world network reduces to a random network, and the average shortest path reaches the smallest, i.e., message propagation speed reaches the fastest. In this paper, we simulate the Bitcoin network by using a random network. Therefore, the transaction propagation efficiency of a random network is the upper bound of the Bitcoin network.

## 3. TRANSACTION PROPAGATION IN BITCOIN NETWORK

### 3.1. Transaction propagation process

Bitcoin is an electronic transaction system running on a P2P network. Nodes jointly maintain a distributed database of replicated ledgers that record all valid transactions. At some fixed interval, e.g., about 10 minutes, nodes in the network synchronize data through blocks. Each block relies on its parent block to guarantee the security and integrity of the previous block. Miners record transactions in blocks, and once there are enough blocks added after a block, those transactions are almost impossible to deny. It is a widely accepted consensus that after six blocks, the transactions are safe. A transaction takes parent transactions as *inputs* and uses all the *inputs* for *outputs*. In general, miners' compensation comes from the unspecified *outputs* and the new block rewards.

During transaction propagation, nodes will verify received transactions, e.g., checking whether *inputs* and *outputs* are valid, before forwarding a transaction to other nodes. In some cases, nodes may receive a transaction whose parent transactions have not arrived yet, so this transaction is called an *orphan transaction* and stored in the orphan transaction pool. An orphan transaction will not be verified and forwarded until all its parent transactions arrive, thus causing transaction propagation delay. A same coin being used multiple times is another potential reason the transaction cannot reach the entire network in time [44], i.e., the double-spending attack. It is impossible to agree on subsequent transactions that depend on conflicting transactions, due to the inconsistent sequence of transactions received by nodes in the Bitcoin network.

There are mainly three kinds of messages during transaction propagation, i.e., *inv*, *getdata*, and *tx*. Here, a node can send *inv* message to declare inventory (transactions or blocks) they possessed, send *getdata* to request missing transactions, and send *tx* to transmit a set of transactions in response to *getdata* message received from the other nodes. We assume that each message contains only one transaction. In Figure 1, we depict the propagation process of a transaction between two nodes A and B. $T_{inv}$ is the time node B receives *inv* from A, $T_{getdata}$ is the time node B sends *getdata* signal to A, $T_{tx}$ is the time node B receives transaction *tx*, $T_{check}$ is the time when transaction verification is completed and *inv* signal is sent. Here, $T_r = T_{tx} - T_{getdata}$ is the round-trip time (RTT) including sending

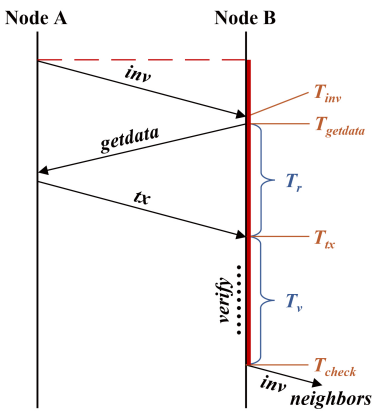



**FIGURE 1.** A transaction propagates between node A and B. The bold red line is the delay time caused by B during transaction propagation. Node A sends *inv* to B to declare transactions, then B sends *getdata* to A to request the transaction. After B receives transaction *tx* from A, B verifies *tx* and sends *inv* to its neighbors

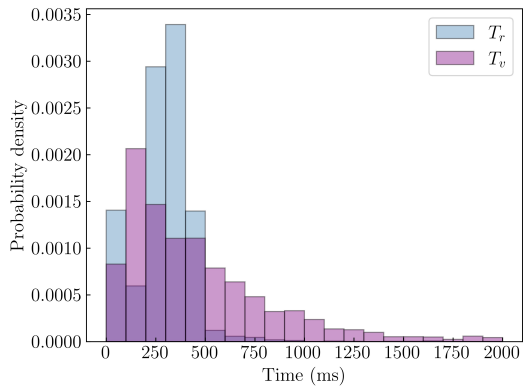



**FIGURE 2.** Distribution of $T_r$ and $T_v$ over 366 blocks collected from Bitcoin Core

*getdata* and *tx*, and $T_v = T_{check} - T_{tx}$ is the time cost for transaction verification. Transaction propagation process between two nodes A and B (we call it a *single-hop*) is $inv(A) \rightarrow getdata(B) \rightarrow tx(A) \rightarrow verify(B)$.

### 3.2. Data collection and analysis

To evaluate transaction propagation efficiency in the existing Bitcoin network, we used Wireshark [45] software to monitor nodes in Bitcoin Core by collecting 366 blocks with block heights from 632,157 to 632,224, 638,079 to 638,249, and 638,271 to 638,400, respectively. Note that these blocks are not continuous because we collected the data in three different time, where each collection took about 24 hours. Our analysis focuses on the transaction propagation process of these blocks, and we correctly identified a total of 27,265 transactions in these 366 blocks. The data underlying this paper will be shared on reasonable request to the corresponding author. The data collecting and processing procedure are as follows.

(1) We use Wireshark to collect transactions in Bitcoin Core and parse the collected data into JSON format.
(2) We analyze the JSON data, sort the data according to the type of requests and then sort them on timestamps in ascending order, and get a list of messages of *getdata*, *inv*, and *tx*, respectively.
(3) We traverse all *getdata* sent by the nodes in Bitcoin Core. Specifically, for all *getdata* messages we perform the following procedures:
    (3.1) Search the list of *inv* messages, find the first received *inv*, get $T_{inv}$ of the transaction, then find the first *inv* sent by this node and get $T_{check}$ accordingly.
    (3.2) Search the list of *tx* messages, find the first received *tx*, and get $T_{tx}$.
    (3.3) Get $T_r$ and $T_v$, where $T_r = T_{tx} - T_{getdata}$ and $T_v = T_{check} - T_{tx}$.

When collecting the 366 blocks, we find that the number of a local node's neighbors lies between $[20, 60]$, stabilizing between $[40, 50]$, which means that the node degree in the Bitcoin network is very large [19].

In Figure 2, we depict the length distribution of $T_r$ and $T_v$, where the average of $T_r$ and $T_v$ is 299.94ms and 661.19ms, respectively. Here, $T_r$ is around 300ms (Note that most of the ping time is also around 300ms in our collected data), and the standard deviation of $T_r$ and $T_v$ is 0.1853ms and 4.82ms, respectively. It means that message transmission time is relatively stable, whereas transaction verification time is not. We also collected a similar amount of Bitcoin Core data on multiple devices at different times, and the distribution of $T_r$ and $T_v$ remains the same as in Figure 2.

In addition, according to our analysis, the average size of message *inv*, *getdata*, and *tx* is 406.32, 260.24, and 443.63 bytes, respectively. Given that network bandwidth is much larger than the size of the above messages, we do not single out the transmission time of a message. Instead, we use round-trip time (RTT) to denote the time used to transmit message *getdata* and *inv*. In other words, the ping time between two nodes overwhelmingly dominates the message transmission time. The time interval between *inv* and *getdata*, and between *getdata* and *tx* are both negligible. For example, according to our analysis of the collected data, $T_{getdata} - T_{inv}$ is less than 1ms. Hence, we ignore them for ease of discussion. Suppose that the time to transmit a message between two nodes is $T_p$, we have $T_p = \frac{T_r}{2}$. The transmission time of message *inv*, *getdata* and *tx* between two nodes can also be regarded as $T_p$. Hence, from Figure 1 we can see that single-hop transaction propagation time is $3 \times T_p + T_v$, where $T_v$ is mainly affected by transactions in *inputs*, i.e., whether the parent transactions can arrive in time.

Another observation is that $T_v$ will increase significantly when a node provides the *synchronization service* to other nodes. The reason is that there are excessive messages waiting to be processed in the local queue rather than the network bandwidth. Therefore, to increase transaction propagation efficiency, we may resort to some possible options given below:

(1) Minimize network delay between nodes by improving network transmission speed.
(2) Minimize transaction verification time by reducing the number of parent transactions required, and broadcasting the transaction immediately.
(3) Minimize the number of redundant messages to save processing time.

Option (1) is orthogonal to our work in this paper, because various factors contribute to network delay, e.g., network bandwidth and topology. Hence, we focus on option (2) and (3), i.e., minimizing transaction verification time and message redundancy. To this end, we propose a novel P2P network structure called local clique network (LCN), and design two efficient transaction propagation strategies in the next section.

## 4. LOCAL CLIQUE NETWORK

### 4.1. The structure of LCN

As mentioned in Section 2, the Bitcoin network has a very large average clustering coefficient [43]. A sizeable average clustering coefficient means that there is a higher possibility that some nodes form a *clique*, i.e., a complete graph. Given a collection of Bitcoin nodes, we construct an LCN by (logically) partitioning these nodes into multiple cliques. Transactions or messages propagate in LCN.

In LCN, each node belongs to exactly one clique, and a node connects to nodes in the inter-cliques. If there is an edge between any two nodes, then these two nodes are *neighbors*. In addition, if any two neighboring nodes are in the intra-clique, they call each other an *insider* with respect to their clique. Otherwise, they regard each other as an *outsider*. Note that a node randomly connects to nodes in inter-cliques (following the same strategy for nodes connecting in a random network), i.e., outsider relationship is random. To illustrate, we give an example of LCN in Figure 3.

In Bitcoin Core, there are some long-running nodes called *seed nodes*. A new node get neighboring nodes by sending *getaddr* to seed nodes. In LCN, a new node joins the network using the above procedure. Specifically, Algorithm 1 shows the process. The new node first requests some nodes $S$ from seed nodes and then requests neighbors of these nodes, which return their insider nodes list $inList$ and outsider nodes list $outList$. Upon receipt of $S$, $inList$ and $outList$, the new node randomly selects a clique to join. For the nodes in the intra-clique, the new node needs to connect to each of them. Then, the new node randomly selects some nodes in inter-cliques as outsiders to connect with.

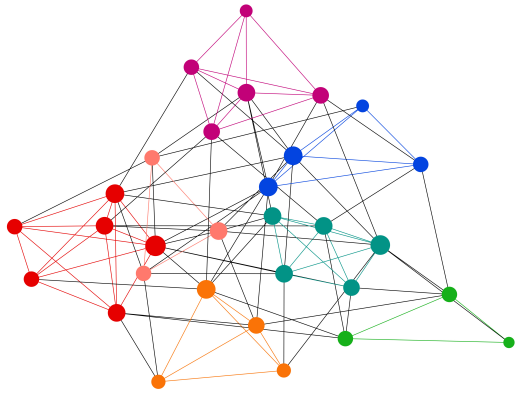


**FIGURE 3.** An example local clique network, where the number of nodes is $N = 30$, the average degree of nodes is $k = 7$, and the average size of cliques is $c = 4$. Here, node size represents magnitude of node degree, and node color distinguishes different clique they belong to, i.e., the edges between nodes in the intra-clique are of the same color. The black edges connect nodes from different cliques

**Algorithm 1:** A new node joining LCN

**Initialize** $C = \emptyset; C' = \emptyset; S = \emptyset$;
$S \leftarrow$ request some nodes from seed nodes;
**foreach** $node \in S$ **do**
  $inList, outList \leftarrow$ request the neighbors of $node$;
  $C \leftarrow C \cup inList$;
  $C' \leftarrow C' \cup inList \cup outList$;
**end**
$insiders \leftarrow$ randomly select a clique from $C$ to join;
$C' \leftarrow C' \backslash insiders$;
$outsiders \leftarrow$ randomly select nodes in $C'$ to connect;

A node is mandatory to join only one clique to avoid some problems. For example, suppose node A and node B belong to both cliques $C_1$ and $C_2$. When A and B receive a message from $C_1$, they need to decide which node forwards this message to $C_2$. This problem worsens when the number of cliques is large, and many nodes join in more than one clique.

If a node receives a message sent by its insider, it only forwards the message to its outsiders. Meanwhile, if a message comes from an outsider, the node forwards the message to all its neighbors. To improve propagation efficiency, we design two message propagation strategies for LCN, and details are given in Section 4.3.

### 4.2. Message propagation redundancy

When a node receives a transaction from a neighbor, the node does not know whether the other neighbors own this transaction, so it sends *inv* to all neighbors, causing an excessive number of *inv* messages.

To quantify propagation redundancy in the network, we define message propagation redundancy (MPR) as the average number of the same message received per node. Let $msg(v_i)$ be the times a node $v_i$ receives the message, MPR is defined as follows:

$$MPR = \frac{1}{N}\sum_{i=1}^{N} msg(v_i) \tag{1}$$

where $N$ is the total number of nodes in the network. When $MPR = 1$, there is no message redundancy during propagation, whereas when $MPR > 1$, there is some degree of message redundancy. Our goal is to reduce message redundancy as much as possible during propagation to improve transaction speed in Bitcoin.

Given a Bitcoin network $BTC = \langle N, M\rangle$, where $N$ is the number of nodes and $M$ is the number of edges. Every two nodes in $BTC$ are connected with a probability $p$, $0 < p < 1$, and the average degree of nodes is $k = \frac{2M}{N}$. Assuming $k > lnN$, then $BTC$ is a connected graph [46]. A message will pass through all the edges in the Bitcoin network, i.e., the message propagates $M$ times. Hence, MPR of Bitcoin network $BTC$ is

$$MPR_{BTC} = \frac{M}{N} = \frac{k}{2} \tag{2}$$

Let $LCN = \langle N, M\rangle$ be a local clique network, $k$ the average node degree, $c$ the average number of nodes in each clique, and $c'$ the average number of outsiders of each node. The total number of cliques is approximately $g = \frac{N}{c}$. Hence, each node belongs to a clique with probability $p = \frac{1}{g}$. The distribution $P(c)$ of the average size of cliques is a binomial distribution, which can be approximated by a Poisson distribution as follows:

$$P(c) = \binom{N}{c} p^c (1-p)^{N-c} \approx \frac{\lambda^c}{c!} e^{-\lambda} \tag{3}$$

where $\lambda = Np = \frac{N}{g}$, and $c$ is the clique size.

Assume that the size of a clique is $c$ and the total number of edges in the clique is $m_c$, then we have

$$m_c = \sum_{i=1}^{c} (c-i) = \frac{c(c-1)}{2} \tag{4}$$

From the above Eq.(3) and Eq.(4), we can get the total number of edges $M_c$ over all cliques as follows:

$$M_c = \sum_{c=1}^{N} m_c P(c) g = \frac{1}{2} g \lambda^2 = \frac{N \times c}{2} \tag{5}$$

Therefore, the average degree of insiders is $\frac{2M_c}{N} = c$, and the average degree of outsiders is $k - c$. Assuming the number of edges not in the cliques is $M_{c'}$, then we have

$$M_{c'} = M - M_c = \frac{N(k-c)}{2} \tag{6}$$

When a message propagates in LCN, there is no message redundancy for insiders, i.e., a total of $g \times (c-1)$ times of propagations in LCN. Note that the message propagation procedure between outsiders is the same as the propagation procedure in the Bitcoin network, where the total number of message propagation is $M_{c'}$. Hence, we can calculate the MPR of our LCN by the following equation:

$$MPR_{LCN} = \frac{g \times (c-1) + M_{c'}}{N} = \frac{k-c}{2} - \frac{1}{c} + 1 \tag{7}$$

Compared to message propagation redundancy of Bitcoin network, i.e., $MPR_{BTC} = \frac{k}{2}$ as given in Eq.(2), the MPR of our LCN is reduced by $\frac{c}{2} + \frac{1}{c} - 1$.

### 4.3. Transaction propagation strategies

When a node receives a transaction in the Bitcoin network, it cannot forward the transaction to other nodes directly, because this may cause a broadcast storm. Although the Bitcoin network itself suffers from the broadcast storm problem, prohibiting messages from direct-forwarding could reduce this risk. Generally, a node periodically declares its newly received transactions through the *inv* message. If a neighbor notices that some transactions are missing, it broadcasts a *getdata* message containing the hash values of the transactions. Then, the nodes that possess those transactions send back the transactions via the *tx* message.

In LCN, nodes adopt different forwarding strategies for insiders and outsiders to improve transaction propagation efficiency, reduce message redundancy, and avoid broadcast storm. Specifically, for insider and outsider nodes, we design the following two message propagation strategies:

(a) When a node receives a transaction *tx*, it sends *inv* immediately to its outsiders, and then it starts to verify *tx*. Upon successful verification of *tx* and receiving *getdata* from an outsider, the node sends *tx* to the outsider.

(b) When a node receives *inv*, it immediately forwards *inv* to its insiders. Once the node receives *tx* from the sender and *getdata* from an insider, it sends *tx* to the insider.

In Section 3, we have introduced single-hop propagation delay, defined the time spent in propagating a single message and the round-trip propagation time as $T_p$ and $T_r$, respectively. Suppose the completion time of verifying a transaction for a sender and a receiver are $T_{check'}$ and $T_{check}$, respectively, and the single-hop propagation delay between them is $T_{check} - T_{check'}$. Figure 4 illustrates the transaction propagation process in

**Algorithm 2:** Propagation strategy (a)

```
tx ← receive a new transaction from a neighbor;
foreach node ∈ outsiders do
    send inv to node;
end
txIsValid ← verify the received transaction tx;
foreach outsider who sends getdata request do
    if txIsValid then
        send tx to the requesting outsider;
    end
end
```

**Algorithm 3:** Propagation strategy (b)

```
inv ← receive a new inventory from a neighbor;
foreach node ∈ insiders do
    send inv to node;
end
send getdata to the neighbor;
tx ← receive a transaction;
foreach insider who sends getdata request do
    send tx to the requesting insider;
end
```

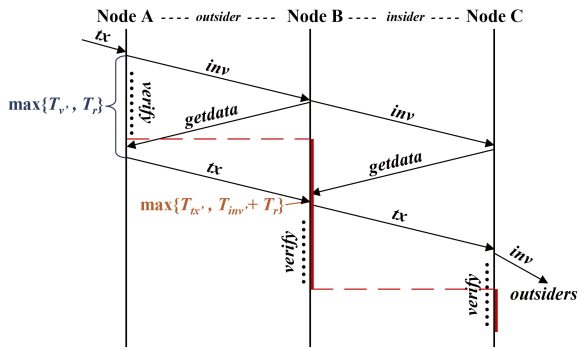
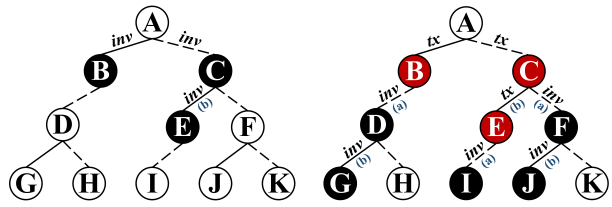



**FIGURE 4.** An illustration of transaction propagation strategies, where node A and B belong to different cliques but are neighbors, and node B and C are in the intra-clique. When A receives $tx$, it immediately sends $inv$ to B before verifying $tx$. After A verified $tx$ and received $getdata$ from B, A sends $tx$ to B. Meanwhile, B directly forwards the received $inv$ to C. After B received $tx$ from A and $getdata$ from C, B sends the $tx$ to C, and sends $inv$ to its outsider




(a) Node A first sends $inv$ to its neighbors B and C

(b) After node B and C send $getdata$ to A

**FIGURE 5.** Node A propagates a transaction to LCN (insiders are connected with solid lines and outsiders with dashed lines), then red nodes B, C, and E verify $tx$. Note that we omit the process of $getdata$ and some nodes for brevity

LCN, where node A and node B use propagation strategy (a) to synchronize transactions. Suppose A and B spend $T_{v'}$ and $T_v$ to verify $tx$, respectively, then the single-hop propagation delay between node A and B is $\max\{T_{v'}, T_r\} - T_{v'} + T_p + T_v$. Nodes B and C belong to the same clique, so we use propagation strategy (b) to synchronize transactions between the two nodes. Suppose B and C spend $T_{v'}$ and $T_v$ respectively to verify $tx$, and node B receives $inv$ and $tx$ at $T_{inv'}$ and $T_{tx'}$, respectively. Then the single-hop propagation delay between B and C is $\max\{T_{tx'}, T_{inv'} + T_r\} - T_{tx'} - T_{v'} + T_p + T_v$.

According to the definition of LCN, a clique is essentially a complete graph, meaning if a node receives a message (either $inv$ or $tx$) sent from some node in a same clique, then the rest nodes in the same clique will also receive this message. Therefore, if a message comes from an insider, the node who receives the message only needs to perform propagation strategy (a). On the other hand, if a message comes from an outsider, the node who receives the message will perform both propagation strategies (a) and (b). In essence, the Bitcoin network is a special case of LCN when $c = 0$, i.e., each node forms a clique, and there is no insider node. Therefore, LCN and the existing Bitcoin network are compatible. The neighboring nodes in the Bitcoin network are outsiders, so nodes can directly use our propagation strategy (a). Note that block propagation can also apply to our message propagation strategies.

### 4.4. Security of LCN

We discuss security issues of LCN in this section. As a public blockchain, one can join the Bitcoin network (or local clique network) freely, so we assume that the adversary may be either a hacker or an autonomous system (AS) that control multiple 'normal' nodes. Nodes in the Bitcoin Core can punish those nodes with abnormal traffic and will not forward illegal transactions (or orphan transactions) in the network, therefore reducing the risk of broadcast storm. On the other hand, at the network level there exists partitioning attacks [47,48] for the Bitcoin network. We discuss the above two security issues that LCN may encounter below.

**Broadcast Storm.** For a P2P network, a broadcast storm is a surge of broadcast traffic. Broadcast storm consumes so many network resources that the network becomes unable to transmit normal traffic, causing the network to be unresponsive. In LCN, propagation strategy (b) does not verify transactions, and may suffer from broadcast storm attack. However, as we will show

below that LCN can resist this potential threat, even though the adversary tries to exploit this 'vulnerability' to implement a broadcast storm attack.

Figure 5 exemplifies the process of node A sending a transaction in LCN. Node A first sends *inv* to all its neighbors, i.e., node B and C, and node C directly forward *inv* to its insider E. Upon receipt of *tx* from A, B employs propagation strategy (a) (because A and B are insiders) to send *inv* to node D, who then sends *inv* to its insiders, i.e., node G. Since A and C are outsiders, C forwards *tx* to its insider E. Meanwhile, C also sends *inv* to its outsider F. Since B, C, and E will verify *tx*, they will stop sending *tx* to their outsiders, i.e., node D, F and I, if *tx* is an invalid transaction. Since propagation strategy (a) used by outsiders will verify each transaction before sending, invalid transactions will not be forwarded unlimitedly, thus avoiding broadcast storm attack.

**Partitioning Attack.** Partitioning attack is a critical security threat, that exploits network connectivity. In terms of spatial partitioning, the adversary aims to isolate a group of nodes to prevent those nodes from generating transactions or receiving blocks. On the other hand, for temporal partitioning, the adversary subverts a group of nodes by feeding them counterfeit blocks and creating forks in the network. In short, in spatio-temporal partitioning, the adversary ensures that the fork is maintained for a long time, thus facilitating a double-spending attack. Basically, a network with low-connectivity is prone to partitioning attack, hence network connectivity is critical to defending against such attack.

In general, P2P network is highly dynamic, i.e., nodes joining and leaving LCN all the time. When a node leaves LCN, it will delete all its intra-clique and inter-clique edges, meaning that the clique the node lies in will remain to be a complete graph and all the inter-cliques also stay completely connected. Meanwhile, Algorithm 1 guarantees that each clique in LCN remains to be a complete graph when a new node joins LCN. Essentially, LCN can be represented by a graph $G = \langle N, M \rangle$, where $N$ and $M$ are nodes and edges, respectively. Assuming $k$ is the average node degree, $c$ the average size of cliques, and $\frac{N}{c}$ the number of cliques, from Eq.(6) we can see that the average node degree of outsiders is $k - c$. Based on random graph theory [46], network connectivity of LCN can be summarized below:

- In the best case, all cliques are complete graphs, i.e., the insiders of each clique are connected. Therefore, we can treat each clique as a node, and there are $\frac{N}{c}$ nodes in total with average node degree $c \times (k - c)$. As long as $c \times (k - c) > ln\frac{N}{c}$ holds, LCN remains connected.
- When a clique in LCN is not a complete graph, i.e., propagation strategy (b) fails for some insiders in the clique, then these insiders can only rely on propagation strategy (a). Consider the worst case where the insiders are not fully connected in each clique, the network remains connected if $(k - c) > lnN$.

From the above discussion we can see that LCN is secure against broadcast storm and partitioning attack, and a message will eventually propagate to the whole network, as long as LCN remains connected.

## 5. EXPERIMENTAL EVALUATION

### 5.1. Experimental setting

In this section, we empirically evaluate the performance of LCN and the proposed propagation strategies. Specifically, we build a Bitcoin network and a local clique network using igraph-python (python 3.8.2). The number of nodes $N$ is set to $10,000$ for both networks. For brevity, we use BTC and BTC(a) to denote the original Bitcoin network and the Bitcoin network with our propagation strategy (a), respectively, and LCN stands for local clique network using propagation strategy (a) and (b). All experiments are conducted on a PC with Intel Xeon E5-2690v3 CPU with 128GB memory, running on Windows 10. Each reported measurement in the results is an average over 100 trails.

### 5.2. Transaction propagation efficiency under ideal settings

Bitcoin network is a complex system in real applications, where transaction transmission and verification time vary significantly. Without loss of generality, we simplify experimental settings and assume some ideal cases. Specifically, we assume that network delay $T_p$ between any two nodes is the same, transaction verification time of a node is $T_v$, $T_v > T_r$, and $T_v \gg T_p$. Therefore, the single-hop propagation delay $\max\{T_{tx'}, T_{inv'} + T_r\} - T_{tx'} - T_{v'} + T_p + T_v$ between insiders can be simplified to $T_p$, and the single-hop propagation delay $\max\{T_{v'}, T_r\} - T_{v'} + T_p + T_v$ between outsiders can be reduced to $T_p + T_v$. Here, we regard the number of hops as transaction propagation time. In Section 3, we calculate the single-hop transaction propagation time as $3 \times T_p + T_v$ in BTC, hence each single-hop transaction propagation of BTC(a) is $2 \times T_p$ faster than BTC.

Table 1 presents the transaction propagation time regarding different clique size $c$ when the average node degree $k$ is fixed to 18. From Table 1 we can see that when $c = 0$, the average number of hops in BTC(a) is $3.5175 \times (T_p + T_v)$. As $c$ increases, the average clustering coefficient (CC) grows significantly, whereas the average shortest path (APL) increases slowly. The rationale is that when the size of clique becomes larger, nodes tend to form bigger clusters, causing both the local clustering coefficient and average clustering coefficient to increase. In addition, the transaction transmission hop $T_p$ (i.e., network delay) is constantly increasing as APL becomes larger.

**TABLE 1.** Transaction propagation time versus average clique size $c$, where $k$ is fixed to 18

| $c$ | APL | CC | MPR | $T_p$ | $T_v$ |
|---|---|---|---|---|---|
| 0 | 3.5158 | 0.0018 | 9.0000 | 3.5175 | 3.5175 |
| 2 | 3.5236 | 0.0142 | 7.0730 | 4.8304 | 2.8907 |
| 4 | 3.5460 | 0.0509 | 7.3335 | 4.3649 | 2.6613 |
| 6 | 3.5828 | 0.1126 | 6.6673 | 4.1438 | 2.5160 |
| 7 | 3.6069 | 0.1527 | 6.2429 | 4.1235 | 2.4961 |
| 8 | 3.6351 | 0.1985 | 5.7981 | 4.0773 | 2.4684 |
| **9** | 3.6690 | 0.2506 | 5.3353 | **4.0690** | **2.4532** |
| 10 | 3.7108 | 0.3092 | 4.8612 | 4.0997 | 2.4688 |
| 11 | 3.7648 | 0.3743 | 4.3788 | 4.1098 | 2.4860 |
| 12 | 3.8383 | 0.4454 | 3.8928 | 4.1869 | 2.5325 |
| 13 | 3.9432 | 0.5225 | 3.4051 | 4.2545 | 2.5913 |
| 14 | 4.0951 | 0.6055 | 2.9162 | 4.3794 | 2.6540 |
| 16 | 4.6383 | 0.7899 | 1.9348 | 4.8664 | 2.8958 |

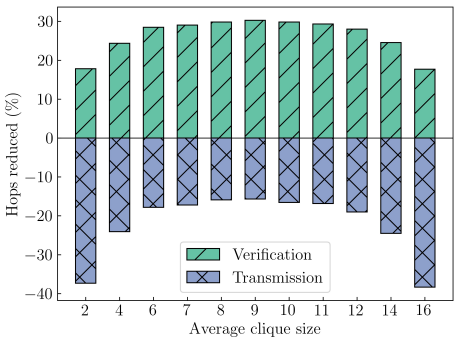



**FIGURE 6.** Assume $k$ is fixed to 18. When a transaction propagates to the whole network, the average number of hops in LCN decreases as compared to BTC. In LCN, the transaction transmission hops $T_p$ increase, but there is a significant decrease in transaction verification hops $T_v$

Figure 6 presents ratio reduction in hops of LCN as compared to BTC(a) in Table 1. It can be seen from Figure 6 that in LCN, the number of transaction transmission hops $T_p$ increases as compared to BTC(a), but the number of verification hops $T_v$ decreases. Since $T_v \gg T_p$, the transaction propagation efficiency of LCN is still better than that of BTC(a). Meanwhile, $T_v$ and $T_p$ first increase and then decrease, achieving the optimal value when $c = \frac{k}{2}$, i.e., $c = 9$, as shown in Table 1 and Figure 6. Therefore, we conclude that the optimal propagation efficiency of LCN is $c = \frac{k}{2}$.

Next, we compare the transaction propagation time of BTC(a) and LCN under different $k$, for which $c = 0$ and $c = \frac{k}{2}$, respectively. From Table 2 we can see that the CC of LCN is much larger than that of BTC(a), and the MPR of LCN is much lower than that of BTC(a). Furthermore, as $k$ increases, the MPR gap between LCN and BTC(a) becomes even larger. For example, when $k = 12$, MPR of LCN decreases by 37.51% compared to MPR of BTC(a), whereas when

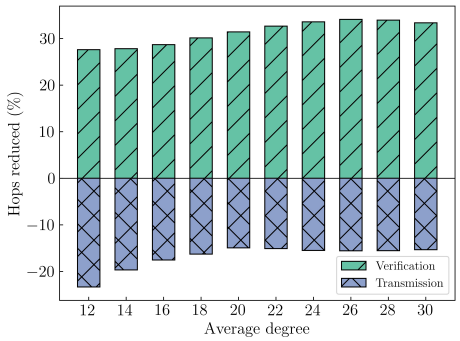



**FIGURE 7.** The number of hops reduced with respect to average node degree $k$ (the higher the value, the better). The average clique size $c$ is $\frac{k}{2}$ in LCN. Compared to BTC with the same average node degree, transaction verification hops $T_v$ in LCN, and the transmission hops $T_p$ also decrease with increasing average node degree

$k = 30$, MPR of LCN shrinks by 44.05%.

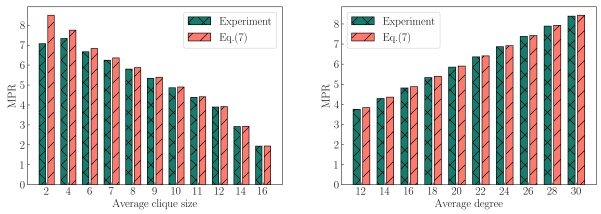
We use $\Delta T_p$ to represent the difference between $T_p$ of BTC(a) and LCN, i.e., $\Delta T_p = BTC(a).T_p - LCN.T_p$. Similarly, we use $\Delta T_v$ to denote the difference between $T_v$ of BTC(a) and LCN, i.e., $\Delta T_v = BTC(a).T_v - LCN.T_v$. It is clear that $\Delta T_p$ and $\Delta T_v$ reflect the reduction in the number of propagation hops of LCN as compared to BTC(a). When $k$ is small, performance gain of LCN over BTC(a) is also small. For example, when $k = 12$, we have $\Delta T_p = -0.9235$ and $\Delta T_v = 1.0920$, i.e., as compared to BTC(a), $LCN.T_p$ increases by 0.9235, whereas $LCN.T_v$ only decreases by 1.0920. However, when $k = 24$, we can see $\Delta T_p = -0.4938$ and $\Delta T_v = 1.0701$, meaning performance improvement of LCN is significant. Figure 7 presents ratio reduction in hops of LCN as compared to BTC(a) in Table 2.

In Figure 7, we can see that that hop decreases as the average degree increases. However, it is interesting that when $k$ becomes larger, say $k = 28$, there is a slight increase in performance gain of LCN. The reason is that when $k = 28$ (Note that this number of average node degree is large enough for a network of 10,000 nodes), LCN has already achieved fast transaction propagation speed by using propagation strategy (a), hence the performance gain by simultaneously employing propagation strategy (b) is not that significant.

We present comparison results of experimentally calculated MPR with the theoretical MPR in Eq.(7) in Figure 8. As shown in Figure 8(a), when $c$ is small, the theoretical MPR in Eq.(7) is significantly larger than the experimentally calculated MPR. However, when $c$ becomes large, the theoretical MPR and the calculated MPR are close. This is because when $c$ is small, the clique size of the network does not completely follow a normal distribution. In addition, as $k$ increases in Figure 8(b), the theoretical MPR and the calculated MPR remain consistent.

**TABLE 2.** Transaction propagation efficiency versus average node degree $k$

| | $k$ | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $c=0$ | **APL** | 3.9659 | 3.7681 | 3.6326 | 3.5158 | 3.4024 | 3.2940 | 3.1954 | 3.1104 | 3.0405 | 2.9853 |
| | **CC** | 0.0012 | 0.0014 | 0.0016 | 0.0018 | 0.0020 | 0.0022 | 0.0024 | 0.0026 | 0.0028 | 0.0030 |
| | **MPR** | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| | $T_p, T_v$ | 3.9566 | 3.7758 | 3.6380 | 3.5074 | 3.4034 | 3.2912 | 3.1869 | 3.1084 | 3.0420 | 2.9883 |
| $c=\frac{k}{2}$ | **APL** | 4.2318 | 3.9786 | 3.7966 | 3.6693 | 3.5645 | 3.4658 | 3.3692 | 3.2774 | 3.1930 | 3.1191 |
| | **CC** | 0.2509 | 0.2507 | 0.2507 | 0.2512 | 0.2509 | 0.2511 | 0.2512 | 0.2516 | 0.2514 | 0.2520 |
| | **MPR** | 3.7494 | 4.2860 | 4.8133 | 5.3316 | 5.8506 | 6.3638 | 6.8737 | 7.3815 | 7.8921 | 8.3921 |
| | $\Delta T_p$ | -0.9235 | -0.7426 | -0.6383 | -0.5711 | -0.5083 | -0.4973 | -0.4938 | -0.4837 | -0.4732 | -0.4582 |
| | $\Delta T_v$ | 1.0919 | 1.0506 | 1.0434 | 1.0569 | 1.0695 | 1.0747 | 1.0701 | 1.0602 | 1.0326 | 0.9974 |




(a) Eq.(7) versus average clique size ($k = 18$)

(b) Eq.(7) versus average node degree ($c = \frac{k}{2}$)

**FIGURE 8.** Experimentally calculated $MPR_{LCN}$ versus theoretical MPR computed by Eq.(7)

In general, the larger the average node degree $k$, the better performance for both LCN and BTC networks. However, a more significant $k$ means that nodes need to process more messages, which may cause congestion in message processing. In the experiment, we can see that LCN has a lower MPR than BTC when the average node degree is the same, and transaction propagation speed in LCN is faster, which can significantly alleviate the congestion problem during message processing.

### 5.3. Transaction propagation efficiency under real settings

This section uses the collected real transaction data in the Bitcoin network to simulate transaction propagation in three networks, i.e., BTC, BTC(a), and LCN. We randomly assign the collected $T_p$ and $T_v$ to each node in the networks. Specifically, when a node receives a message, it uses its own $T_p$ and $T_v$ to calculate the time delay. In this way, our experiment in this section can approximate the transaction propagation process in the real Bitcoin network. Note that we define the propagation time delay as the difference between the time when a node finished transaction verification and the time when the transaction was generated.

Table 3 depicts the comparison results between BTC, BTC(a) and LCN in terms of average propagation time delay (TD) and MPR. As shown in Table 3 and Figure 9, the propagation time delay of BTC(a) is much lower than that of BTC, which means that propagation strategy (a) incurs a remarkable improvement in

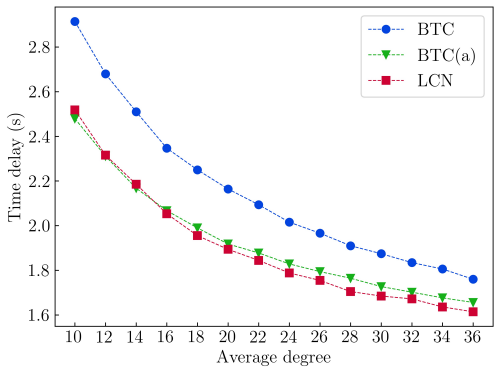



**FIGURE 9.** Average transaction propagation time delay of BTC, BTC(a) and LCN versus average node degree

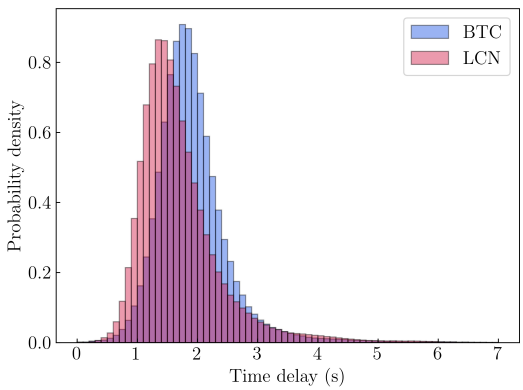



**FIGURE 10.** Distribution of transaction propagation time delay ($N = 10,000, k = 28$)

transaction propagation speed. When $k = 36$, the transaction propagation time delays of BTC(a) and LCN are smaller than that of BTC, i.e., decreases by 5.90% and 8.28%, respectively. When $k = 10$, propagation time delays of BTC(a) and LCN are the smallest as compared to BTC, i.e., decreased by 14.93% and 13.58%, respectively. Meanwhile, from Figure 9 we can see that although when $k < 16$ the propagation delay of BTC(a) is close to LCN, the latter outperforms

**TABLE 3.** Transaction Propagation Time Delay (TD) and MPR Versus $k$

| | Network \ $k$ | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **TD** | **BTC** | 2.91 | 2.68 | 2.51 | 2.35 | 2.25 | 2.16 | 2.09 | 2.02 | 1.97 | 1.91 | 1.87 | 1.83 | 1.81 | 1.76 |
| | **BTC(a)** | 2.48 | 2.31 | 2.17 | 2.07 | 1.99 | 1.92 | 1.88 | 1.83 | 1.80 | 1.76 | 1.73 | 1.70 | 1.68 | 1.66 |
| | **LCN** | 2.52 | 2.32 | 2.19 | 2.05 | 1.96 | 1.89 | 1.84 | 1.79 | 1.76 | 1.71 | 1.68 | 1.67 | 1.64 | 1.61 |
| **MPR** | **BTC,BTC(a)** | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| | **LCN** | 3.21 | 3.75 | 4.28 | 4.81 | 5.33 | 5.85 | 6.36 | 6.88 | 7.39 | 7.89 | 8.40 | 8.91 | 9.41 | 9.91 |

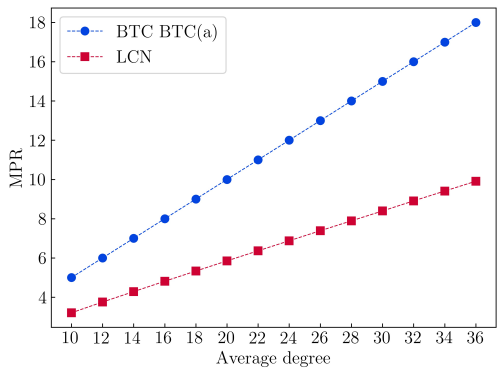



**FIGURE 11.** The average number of *inv* received by nodes in Bitcoin network (BTC) and in local clique network (LCN)

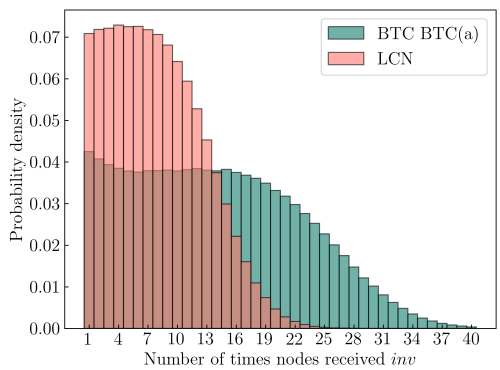



**FIGURE 12.** Distribution of the average number of *inv* received by nodes in BTC and in LCN, respectively ($N = 10,000, k = 28$)

the former consistently when $k \geq 16$.

Figure 10 shows the transaction propagation time delay of nodes in BTC and LCN when $k = 28$, in which the majority of the time delay of LCN lies to the left of BTC. This means that the transaction propagation delay of the majority nodes in LCN is lower than those in BTC.

Since BTC(a) has the same network structure as BTC, the only difference is that BTC(a) uses the propagation strategy (a). Hence, the MPR of BTC and BTC(a) are the same. Figure 11 presents the MPR of *inv* in BTC and BTC(a) with LCN, from which we can see that MPR increases with $k$. However, the growth rate of the MPR in the LCN is much slower than those in BTC and BTC(a). Specifically, when $k = 10$, MPR in LCN is 64.09% of the MPR in BTC, whereas when $k = 36$ the number decreases to 44.95%.

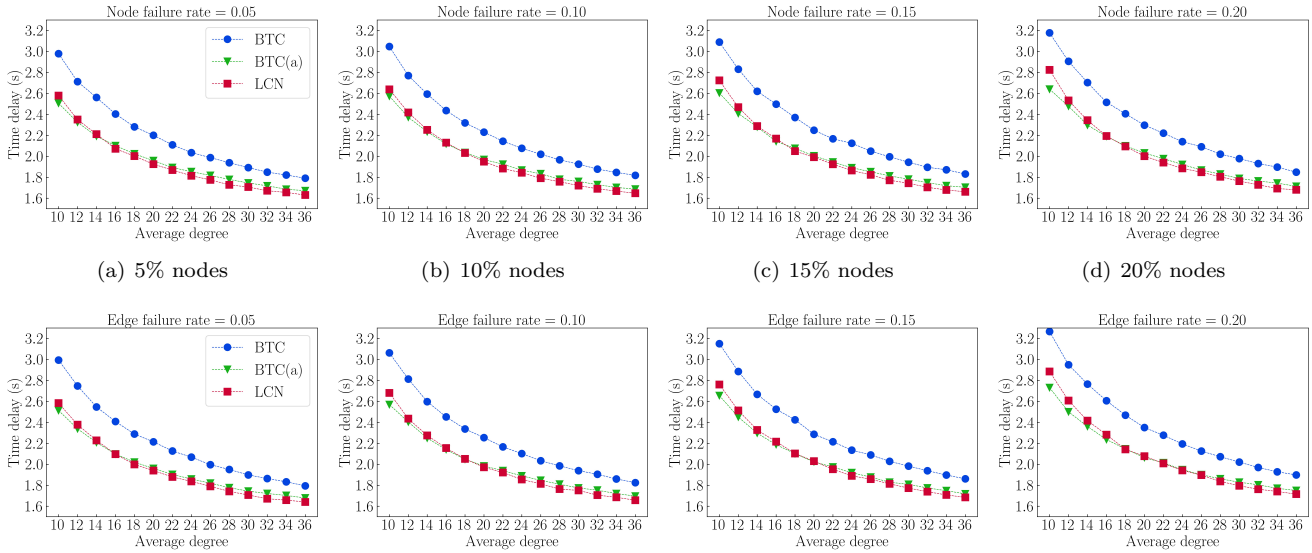
Next, we investigate the average number of times a node receives an *inv* and present the results in Figure 12, where $N = 10,000$ and $k = 28$. In LCN, 90.26% of its nodes receive the same message no more than 14 times, whereas in BTC this number is 54.09%. At the same time, 99.36% of the nodes in LCN receive messages no more than 20 times, as compared to 75.76% of the nodes in BTC. We found that the median number of the *inv* received times is 7 for LCN and 13 for BTC. Therefore, MPR in LCN is much smaller than that in the random network.

From the above experiment results we can see that both random network and LCN benefit from a higher average node degree. However, LCN performs better than random network, in terms of transaction propagation time delay and message propagation redundancy.

### 5.4. Robustness of LCN

In this section, we investigate the robustness of LCN in case of *node failure* (or node exiting the network) and *edge failure*. An edge failure between any two nodes results in unsuccessful transaction propagation between them. Note that if a node finds that any block does not include its transaction, then the node will re-send the transaction. We do not consider this case in our experiment.

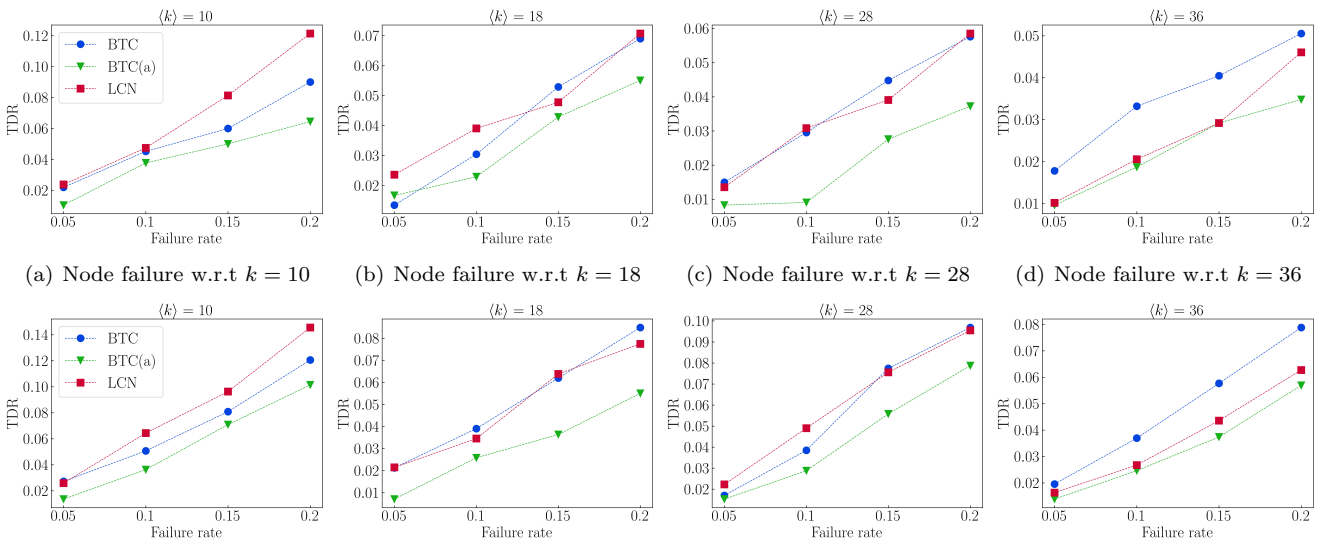
We present the experiment results of the robustness of BTC, BTC(a), LCN versus node failure rate and the average node degree, respectively, in Figure 13. As the node failure rate grows, the transaction time delay of the three networks also increases. Meanwhile, edge failure imposes a greater impact on propagation efficiency in networks than node failure does. For example, when $k = 28$ and failure rate is 20%, node failure causes time delay of 2.0198s, 1.8299s, and 1.8044s for the three networks, respectively. In contrast, edge failure gives raise to the time delay of 2.0718s, 1.8613s, and 1.8366s for BTC, BTC(a), and LCN,




(a) 5% nodes (b) 10% nodes (c) 15% nodes (d) 20% nodes

(e) 5% edges (f) 10% edges (g) 15% edges (h) 20% edges

**FIGURE 13.** The average time delay of a transaction propagating to the whole network versus node/edge failure rate




(a) Node failure w.r.t $k = 10$ (b) Node failure w.r.t $k = 18$ (c) Node failure w.r.t $k = 28$ (d) Node failure w.r.t $k = 36$

(e) Edge failure w.r.t $k = 10$ (f) Edge failure w.r.t $k = 18$ (g) Edge failure w.r.t $k = 28$ (h) Edge failure w.r.t $k = 36$

**FIGURE 14.** TDR versus failure rate and average node degree

respectively. In other words, transaction propagation time delay becomes higher when there are edge failures rather than node failures.

The two network structures, i.e., random network (including BTC and BTC(a)) and LCN, both exhibit good robustness. For example, when $k = 10$, i.e., $k = \lfloor ln(10,000) \rfloor = \lfloor 9.21 \rfloor$, even with 20% edge failure rate, propagation time delays only increase by 9.00%, 6.45%, 12.14% for BTC, BTC(a) and LCN, respectively. In the case of low failure rates, say 5% or 10%, the impact of the node or edge failure on the entire network is relatively small.

To accurately evaluate the impact of different failure rates on transaction propagation time delay, we introduce *time delay rate* (TDR), $TDR = \frac{TD_1 - TD_0}{TD_0}$, where $TD_1$ is the average time delay of a network with node/edge failure and $TD_0$ the average time delay of a network without any node/edge failure. Figure 14 depicts TDR of BTC, BTC(a) and LCN with respect to different failure rates and average node degree $k$.

As can be seen in Figure 14, BTC(a) yields the lowest TDR, which means that node or edge failure imposes the least impact on BTC(a). Although the network structures of BTC(a) and BTC are the same, messages in BTC(a) propagate faster. With small $k$, e.g., $k = 10$, TDR of LCN is higher than that of BTC and BTC(a) when there is either a node or an edge failure. As $k$ increases, however, TDR of LCN gradually decreases

and outperforms BTC.

The average node degree $k$ has a significant impact on the robustness of LCN too. When $k$ becomes large, robustness of LCN improves and approaches to that of BTC(a). For example, when $k = 36$ and the edge failure rate is set to 10%, TDR of BTC, BTC(a) and LCN are 3.69%, 2.46%, and 2.67%, respectively. Nevertheless, the transaction propagation speed in LCN is the fastest, e.g., propagation time delays of BTC, BTC(a), and LCN are 1.8247s, 1.6967s, and 1.6571s, respectively.

The above experiment results show that both random networks and LCN exhibit excellent robustness in case of node/edge failure. When the average node degree is very small, LCN is slightly less robust than BTC. However, when the average node degree becomes large (Note that this is a characteristic of existing Bitcoin network), LCN is more robust than BTC.

## 6. CONCLUSION

In this paper, we focus on the problem of improving transaction propagation efficiency in the Bitcoin network. We conduct an empirical study on real Bitcoin transaction data and find that transaction verification and network delay affect transaction propagation efficiency. We propose a novel P2P network structure called *local clique network* (LCN), and introduce two efficient transaction propagation strategies. Experiment results show that our propagation strategies can significantly reduce transaction propagation delay, and LCN exhibits favorable robustness even with a high node/edge failure rate.

Except for the Bitcoin network, LCN is also suitable for other P2P networks that use the Gossip protocol, as long as the network has characteristics of fast message transmission speed and long single-node verification time. Overall, LCN achieves faster message propagation efficiency and lower message redundancy than random networks. In our future work, we will consider the problem of how to choose the best clique for new nodes to join, so as to further improve transaction propagation efficiency in LCN.

## ACKNOWLEDGEMENTS

This work was supported by NSFC under grant number 62020106013, 61972177, 61932011 and 61877029.

## REFERENCES

[1] Nakamoto, S. (2019) Bitcoin: A peer-to-peer electronic cash system. Technical report.

[2] Underwood, S. (2016) Blockchain beyond bitcoin. *Commun. ACM*, **59**, 15–17.

[3] Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., and Tan, K.-L. (2017) Blockbench: A framework for analyzing private blockchains. *Proceedings of the 2017 ACM International Conference on Management of Data*, New York, NY, USA SIGMOD '17 1085–1100. Association for Computing Machinery.

[4] Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., and Wang, J. (2018) Untangling blockchain: A data processing view of blockchain systems. *IEEE Transactions on Knowledge and Data Engineering*, **30**, 1366–1385.

[5] Dang, H., Dinh, T. T. A., Loghin, D., Chang, E.-C., Lin, Q., and Ooi, B. C. (2019) Towards scaling blockchain systems via sharding. *Proceedings of the 2019 International Conference on Management of Data*, New York, NY, USA SIGMOD '19 123–140. Association for Computing Machinery.

[6] Rahnama, S., Gupta, S., Qadah, T. M., Hellings, J., and Sadoghi, M. (2020) Scalable, resilient, and configurable permissioned blockchain fabric. *Proc. VLDB Endow.*, **13**, 2893–2896.

[7] Zaghloul, E., Li, T., Mutka, M. W., and Ren, J. (2020) Bitcoin and blockchain: Security and privacy. *IEEE Internet of Things Journal*, **7**, 10288–10313.

[8] El-Hindi, M., Binnig, C., Arasu, A., Kossmann, D., and Ramamurthy, R. (2019) Blockchaindb: A shared database on blockchains. *Proc. VLDB Endow.*, **12**, 1597–1609.

[9] Peng, Y., Du, M., Li, F., Cheng, R., and Song, D. (2020) Falcondb: Blockchain-based collaborative database. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA SIGMOD '20 637–652. Association for Computing Machinery.

[10] Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2006) Randomized gossip algorithms. *IEEE Transactions on Information Theory*, **52**, 2508–2530.

[11] David, J. and Thomas, C. (2019) Efficient ddos flood attack detection using dynamic thresholding on flow-based network traffic. *Computers Security*, **82**, 284–295.

[12] Velmurugan, V. and Manickam, J. M. L. (2019) A efficient and reliable communication to reduce broadcast storms in vanet protocol. *Cluster Computing*, **22**, 14099–14105.

[13] Tseng, Y.-C., Ni, S.-Y., Chen, Y.-S., and Sheu, J.-P. (2002) The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, **8**, 153–167.

[14] Wood, G. et al. (2014) Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, **151**, 1–32.

[15] Tao, Y., Li, B., Jiang, J., Ng, H. C., Wang, C., and Li, B. (2020) On sharding open blockchains with smart contracts. *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 1357–1368.

[16] Maymounkov, P. and Mazieres, D. (2002) Kademlia: A peer-to-peer information system based on the xor metric. *Peer-to-Peer Systems*, Berlin, Heidelberg, pp. 53–65. Springer.

[17] Gencer, A. E., Basu, S., Eyal, I., Van Renesse, R., and Sirer, E. G. (2018) Decentralization in bitcoin and ethereum networks. *International Conference on Financial Cryptography and Data Security*, pp. 439–457. Springer.

[18] Zhang, R., Xue, R., and Liu, L. (2019) Security and privacy on blockchain. *ACM Comput. Surv.*, **52**, 1–34.

[19] Decker, C. and Wattenhofer, R. (2013) Information propagation in the bitcoin network. *IEEE P2P 2013 Proceedings*, Trento, Italy, pp. 1–10. IEEE.

[20] Donet Donet, J. A., Pérez-Solà, C., and Herrera-Joancomartí, J. (2014) The bitcoin p2p network. In Böhme, R., Brenner, M., Moore, T., and Smith, M. (eds.), *Financial Cryptography and Data Security*, Berlin, Heidelberg, pp. 87–102. Springer Berlin Heidelberg.

[21] Lischke, M. and Fabian, B. (2016) Analyzing the bitcoin network: The first four years. *Future Internet*, **8**, 7.

[22] Androulaki, E., Karame, G. O., Roeschlin, M., Scherer, T., and Capkun, S. (2013) Evaluating user privacy in bitcoin. In Sadeghi, A.-R. (ed.), *Financial Cryptography and Data Security*, Berlin, Heidelberg, pp. 34–51. Springer Berlin Heidelberg.

[23] Sompolinsky, Y. and Zohar, A. (2015) Secure high-rate transaction processing in bitcoin. *International Conference on Financial Cryptography and Data Security*, pp. 507–527. Springer.

[24] Ruan, P., Chen, G., Dinh, T. T. A., Lin, Q., Ooi, B. C., and Zhang, M. (2019) Fine-grained, secure and efficient data provenance on blockchain systems. *Proceedings of the VLDB Endowment*, **12**, 975–988.

[25] Otsuki, K., Aoki, Y., Banno, R., and Shudo, K. (2019) Effects of a simple relay network on the bitcoin network. *Proceedings of the Asian Internet Engineering Conference*, New York, NY, USA AINTEC '19 41–46. Association for Computing Machinery.

[26] Naumenko, G., Maxwell, G., Wuille, P., Fedorova, A., and Beschastnikh, I. (2019) Erlay: Efficient transaction relay for bitcoin. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA CCS '19 817–831. Association for Computing Machinery.

[27] Li, C., Li, P., Zhou, D., Yang, Z., Wu, M., Yang, G., Xu, W., Long, F., and Yao, A. C.-C. (2020) A decentralized blockchain with high throughput and fast confirmation. *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, July, pp. 515–528. USENIX Association.

[28] Han, Y., Li, C., Li, P., Wu, M., Zhou, D., and Long, F. (2020) Shrec: Bandwidth-efficient transaction relay in high-throughput blockchain systems. *Proceedings of the 11th ACM Symposium on Cloud Computing*, New York, NY, USA SoCC '20 238–252. Association for Computing Machinery.

[29] sallal, M. F., Owenson, G., and Adda, M. (2017) Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network. *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2411–2416.

[30] Shankar Kumar, V. S., Lee, J. J., and Hu, Q. (2021) Indf: Efficient transaction publishing in blockchain. *ICC 2021 - IEEE International Conference on Communications*, pp. 1–6.

[31] Feld, S., Schönfeld, M., and Werner, M. (2014) Analyzing the deployment of bitcoin's p2p network under an as-level perspective. *Procedia Computer Science*, **32**, 1121–1126.

[32] Schrijvers, O., Bonneau, J., Boneh, D., and Roughgarden, T. (2016) Incentive compatibility of bitcoin mining pool reward functions. *International Conference on Financial Cryptography and Data Security*, pp. 477–498. Springer.

[33] Liu, X., Wang, W., Niyato, D., Zhao, N., and Wang, P. (2018) Evolutionary game for mining pool selection in blockchain networks. *IEEE Wireless Communications Letters*, **7**, 760–763.

[34] Lewenberg, Y., Bachrach, Y., Sompolinsky, Y., Zohar, A., and Rosenschein, J. S. (2015) Bitcoin mining pools: A cooperative game theoretic analysis. *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 919–927. Citeseer.

[35] Albert, R., Jeong, H., and Barabási, A.-L. (1999) Diameter of the world-wide web. *nature*, **401**, 130–131.

[36] Broido, A. D. and Clauset, A. (2019) Scale-free networks are rare. *Nature communications*, **10**, 1–10.

[37] Babaioff, M., Dobzinski, S., Oren, S., and Zohar, A. (2012) On bitcoin and red balloons. *Proceedings of the 13th ACM conference on electronic commerce*, pp. 56–73.

[38] Bentov, I., Hubácek, P., Moran, T., and Nadler, A. (2017) Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies. *IACR Cryptol. ePrint Arch.*, **2017**, 300.

[39] Zhang, M., Cheng, Y., Deng, X., Wang, B., Xie, J., Yang, Y., and Zhang, J. (2021) Accelerating transactions relay in blockchain networks via reputation. *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pp. 1–10.

[40] Hao, W., Zeng, J., Dai, X., Xiao, J., Hua, Q.-S., Chen, H., Li, K.-C., and Jin, H. (2020) Towards a trust-enhanced blockchain p2p topology for enabling fast and reliable broadcast. *IEEE Transactions on Network and Service Management*, **17**, 904–917.

[41] Imtiaz, M. A., Starobinski, D., Trachtenberg, A., and Younis, N. (2021) Churn in the bitcoin network. *IEEE Transactions on Network and Service Management*, **18**, 1598–1615.

[42] Zhang, H., Feng, C., and Wang, X. (2019) A greedy-based approach of fast transaction broadcasting in bitcoin networks. *Proceedings of the ACM Turing Celebration Conference - China*, New York, NY, USA ACM TURC '19. Association for Computing Machinery.

[43] Watts, D. J. and Strogatz, S. H. (1998) Collective dynamics of 'small-world' networks. *nature*, **393**, 440–442.

[44] Zhang, S. and Lee, J.-H. (2019) Double-spending with a sybil attack in the bitcoin decentralized network. *IEEE Transactions on Industrial Informatics*, **15**, 5715–5722.

[45] Sanders, C. (2017) *Practical Packet Analysis, 3rd Edition: Using Wireshark to Solve Real-World Network Problems*. No Starch Press.

[46] Erdős, P. and Rényi, A. (1960) On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, **5**, 17–60.

[47] Apostolaki, M., Zohar, A., and Vanbever, L. (2017) Hijacking bitcoin: Routing attacks on cryptocurrencies. *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 375–392.

[48] Tran, M., Choi, I., Moon, G. J., Vu, A. V., and Kang, M. S. (2020) A stealthier partitioning attack against bitcoin peer-to-peer network. *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 894–909.