This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics

An Efficient and Compacted DAG-based Blockchain Protocol for Industrial Internet of Things

Laizhong Cui, Shu Yang, Ziteng Chen, Yi Pan, Mingwei Xu, Ke Xu

***Abstract*—Industrial Internet of Things (IIoT) has been widely used in many fields. Meanwhile, blockchain is considered promising to address the issues of IIoT. However, the current blockchains have limited throughput. In this paper, we devise an efficient and secure blockchain protocol CoDAG based on compacted Directed Acyclic Graph (DAG), where blocks are organized in levels and width. New-generated blocks in CoDAG will be placed appropriately, and point to those in the previous level, making it a well connected channel. Transactions in the network will be confirmed in a deterministic period, and CoDAG keeps a simple data structure at the same time. We also illustrate the attack strategies by adversary, and it is proved that our protocols are resistant to these attacks. Furthermore, we design a CoDAG-based IIoT architecture to improve the efficiency of IIoT system. Experimental results show that CoDAG achieves 164× Bitcoin's throughput and 77× Ethererum's throughput respectively.**

***Index Terms*—Blockchain, Industrial Internet of Things, Compacted Directed Acyclic Graph**

I. Introduction

Recently, industrial Internet of Things (IIoT) is considered a promising technology and has attracted lots of attention. With the development of network communication, the IIoT devices, including sensors, smart objects, etc., will communicate and interact with others via the Internet, allowing IIoT users to control their devices remotely and share their resources efficiently. Nowadays, IIoT has been utilized in many fields, including manufacturing, energy trading, healthcare, vehicles, etc [1]. However, there are some ongoing challenges in IIoT. Firstly, security is a major problem, where the IIoT system may be easily attacked and data may be tampered by malicious third parties. In [2], the authors pointed out that the current IIoT system is vulnerable to resist the Distributed Denial of Service (or *DDoS*) attack. Meanwhile, the traditional IIoT system relies on centralized servers to maintain the network, which is threatened by the weak connectivity and cracked security [3]. Secondly, people are concerned about the privacy, since their data are exposed to network operators and other users. Thirdly, traditional IIoT architecture relies on centralized servers to handle the data computation and storage, and it will hit a bottleneck as the number of IIoT users explodes.

In these years, blockchain has been regarded as a promising solution to the issues of IIoT [4], [5], [6]. Blockchain is a decentralized ledger for distributed parties in the network. Different from traditional system, blockchain provides consensus among distributed participants without any centralized authority. Transactions in the network are organized in blocks, and they are immutable and auditable. Distributed nodes share the ledger, making blockchain a reliable and trustworthy system in the distributed environment. Nowadays, blockchain technology has been employed in IIoT, and many blockchain-based IIoT applications have emerged [7], [8], [9], [10], [11], [12]. Generally, blockchain-based IIoT system has lots of advantages, including:

- blockchain provides reliable services to guarantee the security and robustness of IIoT.
- blockchain protects the sensitive data in IIoT, since only the user with a matched private key gets access to the data.
- users are incentivized to maintain the IIoT network, since blockchain rewards those who make contributions to the system.

Bitcoin [13] was the first blockchain protocol, where blocks in the network are organized in a linear chain. The author proposed the proof-of-work (or *PoW*) mechanism (also called *Nakamoto consensus*) for distributed nodes to achieve consensus. More specifically, miners will compete to solve the cryptographical puzzles, and the winners are allowed to propose new blocks and get the corresponding rewards. Distributed miners in Bitcoin share the ledger, such that the transaction information can not be tampered by any third party.

In a blockchain-based IIoT system, lots of IIoT devices, transactions, and sensitive resources and data are emerging every moment, thus a faster transaction processing and consensus speed among the IIoT devices should be provided. Nevertheless, the current throughput of blockchain is quite low. For example, Bitcoin only achieves 7 transactions per second (or *tps*), and Ethererum [14] only achieves 15 tps

This work has been partially supported by National Key R&D Program of China under Grant No.2018YFB1800302 and No.2018YFB0803405, National Natural Science Foundation of China under Grant No.61772345, No.61625203 and No.61832013, China National Funds for Distinguished Young Scientists under Grant 61825204, Beijing Outstanding Young Scientist Project, and Tencent "Rhinoceros Birds" - Scientific Research Foundation for Young Teachers of Shenzhen University. *(Corresponding author: Shu Yang)*

L.Cui, S.Yang and Z.Chen are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, PR.China.

Y.Pan is with the Department of Computer Science, Georgia State University, Atlanta, USA.

M.Xu and K.Xu are with the Department of Computer Science and Technology, Tsinghua University, Beijing, China and the Beijing National Research Center for Information Science and Technology (BNRist), Beijing, China.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics

[15], and they can not meet the transaction demand of IIoT. To improve the throughput of blockchain, different solutions were proposed, including modification on the linear chain structure (e.g., Bitcoin-NG [16] and GHOST [17]), consensus mechanism (e.g., Algorand [18]), etc. Among them, solutions based on Directed Acyclic Graph (or *DAG*) structure [19], [20], [21], [22], [23] allow multiple blocks to append the network concurrently, thus they achieve the balance between asynchronization and consensus speed. For the traditional DAG-based schemes (e.g., IOTA [19]), however, with more blocks appending the network, new-generated transactions will not be confirmed in a deterministic period, and its security will decline accordingly. Although researchers developed other DAG-based schemes (e.g., Byteball [20], Spectre [21], Phantom [22] and Conflux [23]), their data structures are complicated, which will cause additional and excessive computation to maintain the ledger.

Meanwhile, we point out that the current blockchain-based IIoT architecture is irrational and inefficient. In the current system, every IIoT device is acting as a miner and making computational contribution to the network. However, due to the limited battery, bandwidth, storage and computing power, the IIoT devices can not provide enough computational services for the blockchain maintenance. Besides, many resources of IIoT devices are wasted in the computation process, which will lead to inefficiency of the IIoT system. Hence, a new architecture for IIoT should be developed to make the blockchain-based IIoT system more efficient.

In this paper, we design an efficient and **Co**mpacted **DAG**-based blockchain protocol CoDAG. Unlike the previous DAG-based solutions, we devise a compacted DAG structure with level and width. Blocks in CoDAG are organized in levels, and each level has fixed width which means the maximum value of new-generated blocks in a round. Blocks in the current level will point to those in the previous level, such that CoDAG will develop into a well connected channel. We design algorithms and protocols to place the new-generated blocks at a proper level, making sure that CoDAG has the consistency and liveness property of blockchain. Compared with previous DAG-based schemes, transactions in CoDAG will be confirmed in a deterministic period that relies on the puzzle difficulty. Meanwhile, CoDAG keeps a simple data structure by modifying the existing DAG structure slightly without introducing any additional data structure. Moreover, we demostrate two basic attack strategies by adversaries and a "hybrid" attack strategy that combines the previous strategies, and it is proved that our CoDAG is resistant to them.

Furthermore, we design a CoDAG-based IIoT architecture, where the IIoT devices are divided into *miner*, *gateway* and *node*, and CoDAG is utilized as the backbone data structure and protocols to provide the fast consensus speed. For those IIoT devices that have greater computing power, they will act as miners to maintain the distributed ledger, and their rewards will be the greatest. For the lightweight devices that have better bandwidth, they will act as gateways to maintain the encrypted communication channels and make transactions with others. And other nodes with limited computing power and bandwidth only need to provide the transaction payments and receive the corresponding computational results. In this architecture, we will make full use of the resources in IIoT, and the distributed IIoT devices can achieve fast consensus by using the CoDAG protocols.

To evaluate its performance, we implement a CoDAG prototype which is divided into data layer, chain layer, RPC/Console layer and Dapp layer. Among them, chain layer is the most critical module, which is responsible for maintaining the network communication, data structure, miner management and consensus algorithms. We evaluate our implementation in a real network environment, and the experimental results show that CoDAG scales in width and blocksize. When the blocksize is 2MB and width is 15, the throughput of CoDAG is 1151 tps, which is 164$\times$ Bitcoin's throughput and 77$\times$ Ethererum's throughput respectively. Meanwhile, we study the effect of different values of level formation time (the required duration when a level is full), and simulation results tell us that due to limited computing power and link delay, throughput of CoDAG will stop growing if the level formation time is too small. Furthermore, we investigate the block utilization (the proportion of valid blocks in the network), and it will converge to 90% with the development of network.

The main contributions of this paper are:

- We devise an efficient and secure blockchain protocol CoDAG to meet the transaction demand of IIoT.
- We design a compacted DAG structure with level and width, and propose algorithms to maintain the network.
- We illustrate two basic attack strategies and a "hybrid" strategy in CoDAG, and prove that our CoDAG is resistant to them.
- We propose a CoDAG-based IIoT architecture, where the CoDAG is utilized as the backbone protocols and data structure, and the IIoT devices are divided into different roles.

The rest of paper is organized as follows. Section II will introduce related work of IIoT and blockchain. Section III will present the data structure of CoDAG and the CoDAG-based IIoT architecture. Then the protocols of CoDAG and well connected channel will be studied in Section IV and Section V. Section VI will describe the CoDAG implementation framework in detail. Simulation results will be shown in Section VII. Finally, conclusion and future work will be given in Section VIII.

II. Related Work

A. Blockchain-based IIoT applications

Due to the philosophy of decentralization, blockchain provides secure and reliable services for IIoT. In the blockchain-based IIoT system, data are stored in the distributed ledger, and the data computation process does not rely on any centralized server. Besides, data privacy is guaranteed in blockchain-based IIoT system, making sure that the sensitive industrial data will not be stolen by competitors. Additionally, the IIoT devices sharing their resources (e.g., data storage

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics

TABLE I: Comparison between different consensus protocol

Property / Blockchain	consistency	liveness	transaction speed	confirmation time	full ordered	decentralization
Bitcoin & Ethereum	✓	✓	slow	deterministic	✓	✓
IOTA	✓	✗	fast	non-deterministic	✗	✗
Byteball	✓	✓	slow	deterministic	✓	✗
Spectre	✓	✗	fast	deterministic	✗	✓
Phantom	✓	✓	slow	non-deterministic	✓	✓
Conflux	✓	✓	fast	non-deterministic	✓	✓
CoDAG	✓	✓	fast	deterministic	✓	✓

and computing) will be rewarded, which will encourage IIoT users to contribute to the network. These years have witnessed the emergence of many blockchain-based IIoT applications in academia, and blockchain has been deployed in many industrial fields, including energy trading, smart factory, logistics, etc [24]. In [8], [25], [26], blockchain technology is applied in energy trading and enhances the trading security and privacy. In [9], blockchain combines the healthcare, making the sensitive medical data immutable and traceable. In [27], the authors proposed a decentralized IIoT architecture based on blockchain, and introduced a whitelist mechanism for a smart factory. [28] developed a blockchain-based anonymous reputation system and enhanced the retail marketing. And [29] devised a credit-based PoW mechanism and provided a data authority management for the blockchain-based IIoT system.

However, we note that the current blockchain-based IIoT system utilizes the traditional blockchain platforms (e.g., Bitcoin, smart contracts, etc.), and they can not provide fast consensus for distributed nodes in IIoT system. Besides, the current blockchain-based IIoT system is not efficient enough since lots of computation resources are wasted. Therefore, in this paper, we will discuss the throughput limitation of traditional blockchains, and design a new blockchain protocol CoDAG and a new blockchain-based IIoT architecture.

B. Throughput of current blockchain

In a distributed environment, in order to achieve consensus among nodes, many protocols were proposed, which can be categorized into **permissioned protocol**, **permissionless protocol** and **hybrid protocol**. For permissioned protocols (e.g., PBFT [30], Paxos [31]), only a finite number of designated nodes with permission are selected to maintain the ledger, and its consensus speed will be very fast. Nowadays, permissioned protocols are applied in Hyperledger fabric [32], which allows users to develop their decentralized applications (or *Dapp*) and achieve fast consensus in distributed system. However, in permissioned protocols, only the designated parties are allowed to join the network, which will lead to centralization. For permissionless protocols, Nakamoto consensus [13] is the first blockchain protocol applied in Bitcoin, and nodes can dynamically join or leave the network. The Nakamoto consensus has two properties of blockchain [33], [34]: 1) **consistency**: distributed miners in the network have the consistent view of the chain; 2) **liveness**: a new-submitted transaction will appear in the ledger of all miners. However, the throughput of Bitcoin is only 7 tps, which can not keep up with the demand in IIoT world.

Many improved permissionless protocols were designed to improve the throughput of blockchain, which can be categorized into, 1) **off-chain model** that builds additional chains outside the main chain and reduces the participants in the consensus system (e.g., lightning network [35], pegged sidechain [36]), 2) **on-chain model** that improves and modifies the traditional linear chain structure (e.g., Bitcoin-NG [16], GHOST [17], IOTA [19]). For the off-chain models, lightning network [35] improves the throughput of Bitcoin by building the micropayment channels between nodes. And pegged sidechains [36] is allowed to communicate with the main chain through a two-way peg with desired flexibility. Similarly, other off-chain models rely on the trustworthy third parties, and they compromise decentralization. For the on-chain models, on the one hand, the linear-based solutions including Bitcoin-NG [16], GHOST [17] are developed to improve the efficiency, fairness and usability of Bitcoin. Hovever, in [33], it is proved that linear-based structure is constrained by the maximum network delay in order to guarantee the security of blockchain.

On the other hand, the DAG-based solutions [19]–[23] enjoy a good throughput performance. Unlike the linear-based models, DAG-based solutions allow multiple blocks to attach to the tail of the DAG structure simultaneously. Among them, IOTA [19] provides the fast consensus service for distributed nodes, where the new-generated blocks will confirm and point to two invalidated blocks existed in the network. Meanwhile, there are no miners in IOTA, thus the users can make transactions without any service charge. However, the transaction confirmation time and security of IOTA are not guaranteed as more blocks join the network. Besides, the blocks in IOTA is not full ordered, such that there may exist some conflicting transactions which will not be successfully validated due to the absence of liveness. To solve the issues in IOTA, some improved DAG-based solutions were developed. Byteball [20] maintains the chain by a portion of reputable witnesses elected by others and establishes a full order between blocks. However, it compromises decentralization since the ledger is maintained by a finite number of designated nodes. In the academic area, the DAGlabs proposed two protocols called Spectre [21] and Phantom [22]. Spectre [21] adopts a voting mechanism that every block will submit its vote to a pair of previous blocks and try to determine the block order according to the majority

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics

TABLE II: Notation List

G	graph structure
V	set of nodes
E	set of egdes
l_v	level of node v
w_l	width of level l
K	maximum width of the graph
C_v	connectivity of node v
R_v	reverse connectivity of node v
$\mathcal{C}_l$	candidate set of level l

of votes. Although Spectre has a high throughput and fast transaction confirmation speed, it still can not be extended to support the full linear order over all blocks. To solve the order issue of Spectre, the DAGlabs devised Phantom [22] to support the full order of blockchain maintained by honest nodes. However, Phantom compromises the transaction speed and confirmation time. Conflux [23] introduces the *parent edge* and *reference edge* and maintains the *pivot chain* to keep the full order of blockchain, but its confirmation time is still non-deterministic.

Moreover, Pass *et al.* designed the hybrid protocol [37] that combines the permissioned and permissionless protocols, and devised Thunderella [38] to support instant transactions. However, the performance of hybrid protocol is limited, since the permissionless protocol is still employed as the backbone consensus in hybrid protocol.

As a summary, comparison between different consensus protocols is presented in Table.I.

III. Basic Ideas of CoDAG

A. Data structure of CoDAG

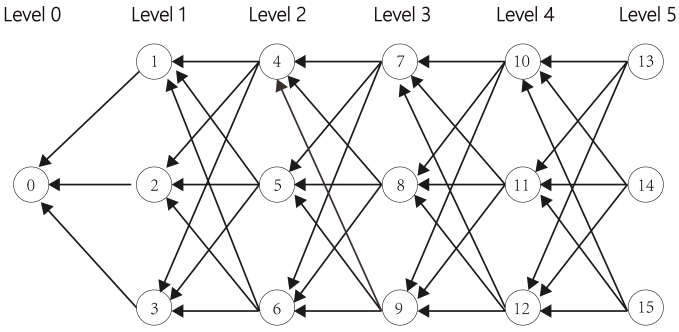


Fig. 1: A simple example for CoDAG

In this paper, we devise a blockchain structure **Co**mpacted **D**irected **A**cyclic **G**raph CoDAG, where the DAG structure is employed as the backbone data structure. The advantage of DAG structure is that, it fits well in asynchronous environment, thus it can achieve high throughput among distributed nodes. Different from previous DAG-based blockchain schemes, CoDAG is more like a channel within fixed width, and blocks are closely connected in adjacent levels. Therefore, as blocks join the network, CoDAG will always develop in a compacted DAG shape.

We call $G = (V, E)$ the graph, where V is the set of blocks or nodes, and E is the set of edges between nodes. We denote l_v as the level of node v, which means the value of longest path from the genesis to it. w_l is denoted as the width of level l, which is constrained by K, that is, for $\forall l$, $w_l < K$. The width also means the maximum value of nodes that can concurrently join the network in a round. It can also be adjusted or self-adaptive by the block generation rate.

In the graph, nodes in current level will point to those in the previous level. To measure how closely a node connects with others, we define *connectivity* $C_{v,w}$ as the number of paths between node v and w. Specifically, the connectivity from node v to itself is 1, that is, $C_{v,v} = 1$. Besides, we define the connectivity of node v as the number of paths from genesis to it, which is denoted as C_v. In Fig.1, $C_8 = 9$, including 9 paths:

$\langle 8{\to}4{\to}1{\to}0\rangle$, $\langle 8{\to}4{\to}2{\to}0\rangle$, $\langle 8{\to}4{\to}3{\to}0\rangle$,
$\langle 8{\to}5{\to}1{\to}0\rangle$, $\langle 8{\to}5{\to}2{\to}0\rangle$, $\langle 8{\to}5{\to}3{\to}0\rangle$,
$\langle 8{\to}6{\to}1{\to}0\rangle$, $\langle 8{\to}6{\to}2{\to}0\rangle$, $\langle 8{\to}6{\to}3{\to}0\rangle$.

In CoDAG, the leaf nodes of the graph are *tips*, and the tips with largest connectivity is denoted as *navigators*. For example, the navigators in Fig.1 are node 13, 14, 15, since they share the value of connectivity. Furthermore, to maintain the structure of CoDAG, we define *reverse connectivity* R_v of node v as the number of paths from navigators to it. In Fig.1, $R_8 = 9$, including 9 paths:

$\langle 13{\to}10{\to}8\rangle$, $\langle 13{\to}11{\to}8\rangle$, $\langle 13{\to}12{\to}8\rangle$,
$\langle 14{\to}10{\to}8\rangle$, $\langle 14{\to}11{\to}8\rangle$, $\langle 14{\to}12{\to}8\rangle$,
$\langle 15{\to}10{\to}8\rangle$, $\langle 15{\to}11{\to}8\rangle$, $\langle 15{\to}12{\to}8\rangle$.

To guarantee the security of CoDAG, the honest nodes in network need to be connected closely and enjoy a higher reverse connectivity for the current navigator. The notations of CoDAG is listed in Table.II.

B. The CoDAG-based IIoT architecture

Generally, the IIoT devices are limited by their battery, computing power, storage, communication bandwidth, etc. In the IIoT network, due to the huge transaction number, the overwhelming transactions can not be handled by IIoT devices, and lots of resources are wasted in the computation process, which will lead to inefficiency of the system. To make full use of the computation and other network resources, we design a framework shown in Fig.2, where the IIoT devices are divided into *miner*, *gateway* and *node*.

- **Miner**: A miner will maintain the ledger using our CoDAG protocols. Similar with Bitcoin, the miners in our architecture will use their computing power to bundle the transactions into blocks and append the blocks to our proposed DAG-based structure using the CoDAG protocols. Miners will get the greatest reward, since their contribution to the network is the most.
- **Gateway**: A gateway can be a lightweight IIoT device with enough bandwidth resources, including 5G base stations connecting other IIoT devices, etc. A gateway is responsible for downloading the part of the ledger, maintaining the encrypted communication channels with other IIoT devices, and making transactions with other gateways.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics

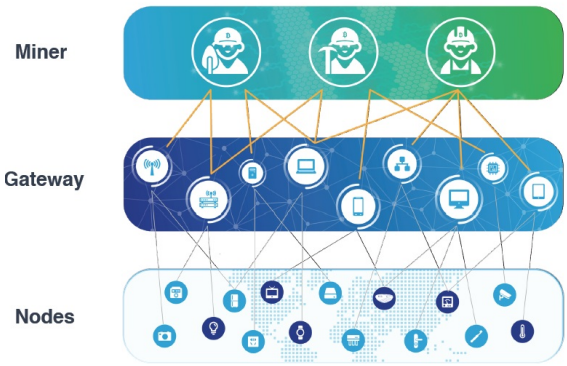


Fig. 2: CoDAG Ledger Framework

- **Node**: A node will submit the payments and other data to the gateway, and receive the corresponding results. If a device has enough computing power or bandwidth, it can be a miner or gateway as well.

In this architecture, CoDAG is utilized as the backbone data structure and protocols, which can provide the fast consensus speed to meet the transaction demand in the IIoT world. Due to the profound throughput of CoDAG, the miners in our CoDAG-based IIoT architecture can achieve a fast transaction processing speed, thus the transaction fee for each payment will be relatively small, and the IIoT devices are more willing to make transactions in the network. Meanwhile, we make full use of the resources of IIoT devices and deploy the lightweight nodes (e.g., the gateway) to help maintain the ledger, which will improve the efficiency of our CoDAG-based IIoT system. In the future, to improve the efficiency further, we will introduce the hybrid consensus to CoDAG, and the system will be divided into *fast mode* and *slow mode*. In the slow mode, miners will run the PoW protocol and compete to join the membership of Byzantine agreement and enter the fast mode, and those who successfully enter the fast mode will achieve the consensus and maintain the ledger more rapidly and get their corresponding rewards.

IV. Protocols of CoDAG

A. Overview

To clarify the protocol of CoDAG, a brief introduction of Nakamoto consensus will be given.

Nakamoto consensus in a nutshell: Nakamoto consensus is applied in Bitcoin and provides the PoW mechanism for distributed nodes to achieve consensus. Participants in the network, called *miners*, will try their best to solve a random math puzzle. The miners who successfully solve the puzzle are allowed to bundle transactions into a block, and append the block to the end of the linear chain, and get the corresponding rewards. However, due to the propagation delay, forks will happen when a block is mined by multiple miners simultaneously. When the forks exist, honest miners will append new blocks to the longest chain.

Protocol of CoDAG: Similar with Nakamoto consensus, in CoDAG, transactions will be enveloped into blocks by miners who successfully solve the cryptographical puzzles. And the miners who successfully mine a new block will get rewarded, thus miners are incentivized to contribute to the network. Due to the network latency, forks may appear where more than K nodes join the same level. In this case, nodes with higher reverse connectivity will have a higher priority compared with other nodes who have a lower reverse connectivity, and the top K nodes in level l with higher reverse connectivity will be defined as *candidates*, and the candidate set is denoted as $\mathcal{C}_l$. A special case is that several nodes with the same priority belong to the borderline candidates, and they will be considered as candidates and share the rewards.

B. Algorithms of CoDAG

Block Generation Algorithm: To compute where a new block should be placed in CoDAG, we design $Generate()$ as the main function (see in Algorithm 2) and $Connectivity()$ (see in Algorithm 1) as the subfunction to compute the connectivity of a node. The input of $Generate()$ is the graph G and the new block x, and the output is the new graph with x and related edges.

In a graph $G = (V, E)$, L represents the maximum level, and $\langle v, w \rangle$ denotes an edge pointing from node v to node w. When a new-generated block x appears and wants to join the network, $Generate()$ (Algorithm 2) will call the $Connectivity()$ (Algorithm 1) to compute the connectivity of every node in G. In $Connectivity()$, we adopt the traversal method level by level iteratively from line 5 to line 9. The principal of $Connectivity()$ is that, for every node v in level l, if there are directed edges connecting node v and nodes in the last level, the algorithm will add the connectivity of nodes in the last level to node v. For example, in Fig.1, suppose we are computing the connectivity of node 8 in l_3, and there are node 4, node 5 and node 6 in l_2. We note that the directed edges $< 8, 4 >$ connects the node 8 and node 4, thus the algorithm will add the connectivity of node 4 to node 8, that is, $C_8 = C_8 + C_4$. Similarly, $Connectivity()$ will execute $C_8 = C_8 + C_5$ and $C_8 = C_8 + C_6$ in l_2, and we will get the connectivity C_8 of node 8. $Connectivity()$ will start from the first level l_0, compute the connectivity of other nodes in the following levels iteratively, and finally return the results to $Generate()$.

In $Generate()$, the node x in level L will be placed according to the connectivity, and point to K nodes that has the largest values of connectivity in level $L-1$. Meanwhile, if level L is full of K nodes, the value of L will increase by 1. Finally, the network information $G = (V, E)$ will be updated in line 6 and 7.

Candidate Selection Algorithm: To deal with forks in CoDAG, we develop $Select()$ (see in Algorithm 3) to select the candidates of a level according to their reverse connectivity. The input of Algorithm 3 is the graph $G = (V, E)$ and level l, and the output is the candidate set in l.

Firstly, from line 2 to line 5, $Select()$ will call $Connectivity()$ to compute the connectivity of each node

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics

Algorithm 1: Connectivity(G, v)

```
1  begin
2      C_v ← 0, ∀v ∈ V;
3      L ← maximum level of G;
4      C_genesis ← 1;
5      for l_v + 1 ≤ l ≤ L do
6          for ∀v ∈ {v ∈ V|l_v = l} do
7              for ∀w ∈ {l_w = l − 1} do
8                  if ⟨v, w⟩ ∈ E then
9                      C_v+ = C_w;
10     return {C_v|v ∈ V};
```

Algorithm 2: Generate($G = (V, E)$, x)

```
1  begin
2      Connectivity(G, genesis);
3      if |{v_L}| = K then
4          L+ = 1;
5      S ← K nodes that have the largest connectivity
         in level L − 1;
6      V ← V ⋃{x};
7      E ← E ⋃{⟨x, v⟩, v ∈ S};
```

Algorithm 3: Select($G = (V, E)$, l)

```
1  begin
2      L ← maximum level of G;
3      C_l ← ∅;
4      Connectivity(G, genesis);
5      N ← {v|C_v ≤ C_w, ∀w ∈ V} # there may be
         multiple nodes with equal connectivity;
6      R_v ← 0, ∀v ∈ V, R_x = 1, ∀x ∈ N;
7      for L − 1 ≥ k ≥ l do
8          for ∀v ∈ {v ∈ V|l_v = k} do
9              for ∀w ∈ {l_w = k + 1} do
10                 if ⟨w, v⟩ ∈ E then
11                     R_v+ = R_w;
12     T ← |{v|l_v = l}| ≤ K;
13     if T ≤ K then
14         return C_l ← T ;
15     else
16         sort(T, {C_v|v ∈ T}) #sort T from highest to
             lowest according to its reverse connectivity
             value ;
17         for 0 ≤ i ≤ K − 1 do
18             C_l ← C_l ⋃{T[i]}
19         return C_l ← C_l ⋃{v|v ∈ T&R_v = R_{T[K−1]}};
```

and choose the navigators that has the largest values of connectivity. Line 7 to line 11 is the computation process of reverse connectivity. Similarly, Algorithm 3 also adopts the traversal method level by level iteratively. Contrary to Algorithm 1, the traverse order of $Select()$ is from the outer level $L-1$ to the inner level, since the reverse connectivity is the value of paths pointing from the navigators. And the reserve connectivity of navigators in maximum level L has the same values of 1. In Fig.1, for example, suppose we are computing the reverse connectivity of node 8 in l_3. Since there are edges $<10,8>, <11,8>, <12,8>$ connecting the nodes between l_3 and l_4, the algorithm will add the reverse connectivity of node 10, node 11 and node 12 to node 8, that is, $R_8 = R_8 + R_{10}$, $R_8 = R_8 + R_{11}$, $R_8 = R_8 + R_{12}$. Finally, line 12 to line 19 is the candidate selection process. If there are no more than K nodes in level l, they will be selected as candidates of this level. Otherwise, $Select()$ will sort and choose the top K nodes that have the highest values of reserve connectivity as the candidates of level l.

V. Well Connected Channel of CoDAG

Different from traditional linear chain, CoDAG employs a compacted DAG-based structure that is more suitable in asynchronous environment, and it will develop into a well connected channel whose width can be adjusted flexibly. To attack the network, adversaries will try to control the navigators, such that the new-generated blocks will point to the adversarial navigators. Therefore, we consider a situation where an attacker builds an adversarial channel with higher connectivity and controls the navigators. Similar with Bitcoin [13], we have the following notations:

- p = proportion of honest computing power
- q = proportion of adversarial computing power
- q_z = probability of an adversary catching up from z levels behind

Lemma 1.

$$q_z = \begin{cases} 1, & \textit{if } p \le q. \\ (q/p)^z, & \textit{otherwise}. \end{cases} \tag{1}$$

If honest and adversarial nodes start to mine the next w blocks simultaneously, the winning rate of honest party is p, and adversary party is q. Hence, the proof of Lemma 1 is the same as [13].

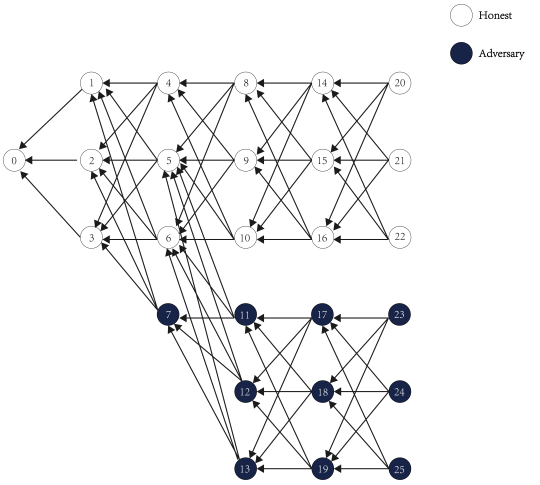
Then we investigate the transaction confirmation time.

Lemma 2. *If node v is z levels away from largest level L, the possibility $\mathcal{P}$ of deprivation of its candidate identity is:* $\mathcal{P} = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - \left(\frac{q}{p}\right)^{z-k}\right)$, *where* $\lambda = z\frac{q}{p}$.

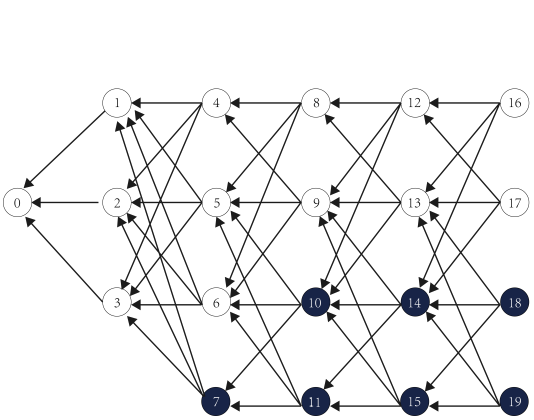
Proof. We suppose there are two nodes in level l, where node a is produced by honest miners and b is produced by adversary miners, and suppose $R_a > R_b$ when the largest level of the graph L satisfies $L - l \le z$.

We first prove that in any level $k > l$, for the two honest nodes v_i and v_j, then $C_{v_i,a} = C_{v_j,a}$ and $C_{v_i,b} = C_{v_j,b}$; for the two adversary nodes w_i and w_j, $C_{w_i,a} = C_{w_j,a}$ and $C_{w_i,b} = C_{w_j,b}$. We prove it through induction. In level $l+1$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics



(a) Adversary builds an alternate channel



(b) Adversary competes in the original channel

Fig. 3: Two attack strategies in CoDAG

all honest nodes will point to a and avoid pointing to b, thus their connectivity to a is 1 and connectivity to b is 0; all adversary nodes will point to b and avoid pointing to a, thus their connectivity to a is 0, and connectivity to b is 1. For a level $k > l$, suppose the nodes in the level satisfy the assumption, and there are P honest nodes and Q adversary nodes, any honest node v^k satisfies $C_{v^k,a} = X$ and $C_{v^k,b} = Y$, any adversary node w^k satisfies $C_{w^k,a} = X'$ and $C_{w^k,b} = Y'$. Then in level $k+1$, for any honest node v^{k+1}, $C_{v^{k+1},a} = X*P+X'*(K-P)$ and $C_{v^{k+1},b} = Y*P+Y'*(K-P)$, for any adversary node w^{k+1}, $C_{w^{k+1},a} = X'*Q+X*(K-Q)$ and $C_{w^{k+1},b} = Y'*Q+Y*(K-Q)$.

Then we prove the above lemma. For a level $k > l$, $\mathcal{R}(k) = \sum_{\forall v^k} R_{v^k}$ is the total reverse connectivity of all honest nodes in level k, and $\hat{\mathcal{R}}(k) = \sum_{\forall v^k} R_{w^k}$ is the total reverse connectivity of all adversary nodes. Suppose $\frac{\hat{\mathcal{R}}(k)}{\mathcal{R}(k)} \geq \epsilon$ should be satisfied to make the adversary party catch up with the honest party. In level $k+1$, we will have $\mathcal{R}(k) = \mathcal{R}(k+1)*P+\hat{\mathcal{R}}(k+1)*(K-Q)$ and $\hat{\mathcal{R}}(k) = \hat{\mathcal{R}}(k+1)*Q+\mathcal{R}(k+1)*(K-P)$ indicating that all honest nodes will point to honest nodes first and all adversary nodes will point to adversary nodes first. Based on the above equations, we can infer that $\frac{\hat{\mathcal{R}}(k+1)}{\mathcal{R}(k+1)} \geq \frac{\epsilon*P+P-K}{Q+Q*\epsilon-\epsilon*K}$. Because P and Q follow two independent poisson distributions, the expectation value $E(\frac{\hat{\mathcal{R}}(k+1)}{\mathcal{R}(k+1)}) \geq \frac{\epsilon*E(P)+E(P)-K}{E(Q)+E(Q)*\epsilon-\epsilon*K} \geq \epsilon * E(\frac{P}{Q}) = \epsilon * \frac{p}{q}$, where $E(\frac{P}{Q})$ is the ratio of the computing power of the honest party to the computing power of the adversary party. □

An example. Suppose there is an adversary trying to conduct a malicious behavior (e.g., double-spending attack [39], [40]). Similar with Bitcoin, the attacker needs to append the blocks that contains his own forged transaction records to the network and control the candidate identity of the CoDAG, such that the following new-generated blocks will verify and point to those adversary candidates.

We will compute the possibility of the candidate identity deprivation of node v in different values of q and z (shown in Table.III).

TABLE III: Possibility of Candidate Identity Deprivation

$q = 0.1$		$q = 0.2$	
$z = 0$	$\mathcal{P} = 1.0000000$	$z = 0$	$\mathcal{P} = 1.0000000$
$z = 1$	$\mathcal{P} = 0.2045873$	$z = 2$	$\mathcal{P} = 0.2039285$
$z = 2$	$\mathcal{P} = 0.0509779$	$z = 4$	$\mathcal{P} = 0.0529978$
$z = 3$	$\mathcal{P} = 0.0131722$	$z = 6$	$\mathcal{P} = 0.0142506$
$z = 4$	$\mathcal{P} = 0.0034552$	$z = 8$	$\mathcal{P} = 0.0038851$
$z = 5$	$\mathcal{P} = 0.0009137$	$z = 10$	$\mathcal{P} = 0.0010670$
$z = 6$	$\mathcal{P} = 0.0002428$	$z = 12$	$\mathcal{P} = 0.0002943$
$z = 7$	$\mathcal{P} = 0.0000647$	$z = 14$	$\mathcal{P} = 0.0000814$
$z = 8$	$\mathcal{P} = 0.0000173$	$z = 16$	$\mathcal{P} = 0.0000226$
$z = 9$	$\mathcal{P} = 0.0000046$	$z = 18$	$\mathcal{P} = 0.0000063$
$z = 10$	$\mathcal{P} = 0.0000012$	$z = 20$	$\mathcal{P} = 0.0000017$

We can see that given a fixed value of q, when the value of z grows, the possibility of the candidate identity deprivation will decrease exponentially. Therefore, as the network develops, our CoDAG protocol is secure to resist the candidate identity deprivation.

We point out that there are two attack strategies that may happen in CoDAG. The first one is, the adversary will build a completely separate new channel (see in Fig.3(a)). In this case, the adversary will point to adversarial nodes in the new channel rather than the honest node in the original channel, which is the same as Bitcoin.

The second one is, the adversaries will compete in the original channel with honest nodes (see in Fig.3(b)). For an adversary, it will first point to adversarial nodes, and the remaining pointers will be given to honest nodes. The strategy for an honest node is just opposite.

In the second strategy, each node has K votes. For the honest nodes, they will firstly give their votes to honest nodes, and then randomly give their remaining votes to adversary nodes in previous level. Contrary to honest party, the adversary nodes will firstly give their votes to adversary nodes, and then randomly give their remaining votes to honest nodes. However, in Lemma 2 it is proven that, if the honest party dominates the computing power, they will get most votes. Meanwhile, as more nodes join the network and number of levels grows, honest votes will get accumulated,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics

and the security of CoDAG will increase further. Hence, CoDAG is secure enough to resist these two attack strategies.

We note that for the same level, if there are $2 \cdot K$ nodes, then the attack strategy belongs to the first one, since the adversary will build an alternate channel whose width is also K. And if there are less than $2 \cdot K$ nodes, the attack strategy belongs to the second one. Moreover, we point out that the adversary will conduct a "hybrid" attack strategy, where the first strategy is used in some levels, and the second strategy is used in other levels. However, our CoDAG protocol is still resistant to the "hybrid" strategy, which can be proved based on the proofs of two basic strategies we mentioned, and the details are omitted in this paper. Therefore, these two basic attack strategies can cover the "hybrid" strategy by adversary, and the security is guaranteed in our CoDAG protocols.

VI. Implementation

A. Implementation Framework

Fig.4 demostrates the framework of our implementation, where the data layer, chain layer, RPC/Console layer and Dapp layer are involved. Among them, chain layer is the core layer in CoDAG, which can be divided into:

- **Network Module:** Synchronize the network information by P2P protocols.
- **CoDAG Module:** Maintain the data structure and algorithms.
- **Miner Module:** Manage the miners and puzzle difficulty.
- **Consensus Algorithm Module:** Maintain the consensus among nodes.

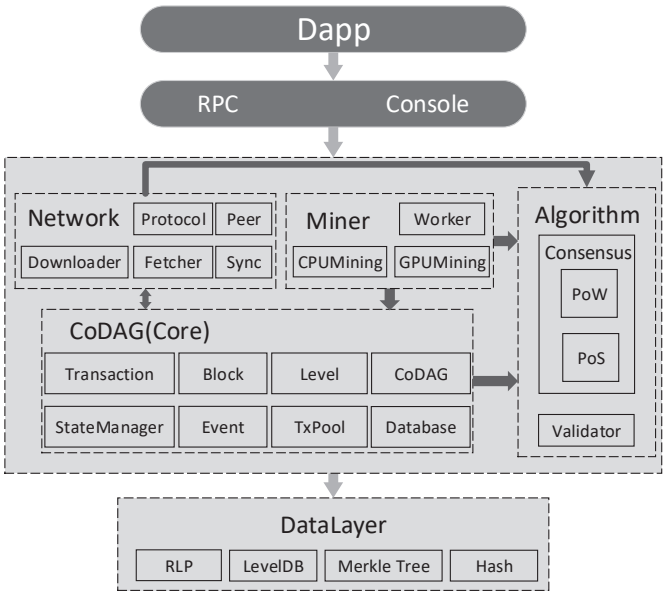


Fig. 4: Implementation Framework

B. Implementation Details

The communication and synchronization process is shown in Algorithm 4. From line 2 to line 4, nodes will build the connections with others by $handshake(peer)$ and choose the $best\_peers$ in good condition. In line 6 and 7, $peer\_nodes$ means the navigators in $peer\_info$, and dag_nodes means the navigators in $peer$. From line 8 to line 9, if any difference is found in the current level, CoDAG will merge the information of dag_nodes and $peer\_nodes$ and synchronize the block information in this level. Then in line 10 and line 11, $peer\_nodes$ and dag_nodes will point to blocks in previous level. The loop from line 8 to line 11 will not stop until $peer\_nodes$ and dag_nodes share the block information at a certain level.

Algorithm 4: Protocol-Start(DAG, $peer\_list$)

```
1  begin
2      for ∀peer ∈ peer_list do
3          peer_info = handshake(peer);
4      best_peers = choose(peer_info);
       // the navigators of the best peers have larger
       level number;
5      for ∀peer ∈ best_peers do
6          peer_nodes = extract_navig(peer_info);
7          dag_nodes =
            extract_level(DAG, level_of(peer_nodes));
           // set to be empty if the level does not exist
8          while compare(peer_nodes,
            dag_nodes)!=true do
               // there exists difference in the current
               level;
9              merge(peer_nodes, dag_nodes);
10             peer_nodes=parent(peer_nodes)
                // RPC request and the peer should
                eliminate the redundancies;
11             dag_nodes=parent(DAG, dag_nodes);
```

The process of inserting a block is given in Algorithm 5. Like Bitcoin, transactions in CoDAG are considered valid after being confirmed in the last $constant$ levels. From line 2 to line 5, for $\forall l$ where $l_v - l < constant$, the algorithm will validate $node$ and its containing transactions in $peer\_node\_pool[l]$. If $node$ is valid, it will join the $node\_pool$. In line 6, $node\_pool$ will be sorted by their connectivity, and the top K nodes with the largest connectivity will become the navigators.

Algorithm 5: Protocol-Insert(v)

```
1  begin
2      for ∀l, where l_v − l < constant do
3          for node in peer_node_pool[l] do
4              if Validate(node) then
5                  node_pool = node_pool ∪ {node};
6      elected_node_pool = Sort(node_pool);
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics

TABLE IV: Default Simulation Parameters

parameter	value
blocksize	128KB
width	10
level formation time	13s
average link delay	50ms

Algorithm 6 shows the process of updating the world state. We define $world\_state$ as the states of participants in the network, where their account balances and transaction records are involved. And we call $stable\_level$ the level where the algorithm stops visiting. We note that some jitters may happen at the beginning stage, thus CoDAG will ignore the first $stable\_level$ levels. From line 2 to line 9, for $\forall l$ where $l_v > stable\_level$, existing transactions in $elected\_node\_pool$ will be visited and added to tx_set. In line 6, the algorithm will sort the transactions in tx_set by their amounts, making sure those transactions with a larger amount can be bundled fast. Finally, CoDAG will execute the tx_set and update the $world\_state$ of the network.

Algorithm 6: Protocol-State(v)

```
1 begin
2     for ∀l, where l_v > stable_level until
        navigator_level do
3         for node in elected_node_pool do
4             for tx in node do
5                 tx_set = tx_set ∪ {tx};
6         tx_set = Sort(tx_set);
7         for tx in tx_set do
8             world_state = Exec(tx);
9         Save(world_state);
```

VII. Evaluation

A. Experiment Setup

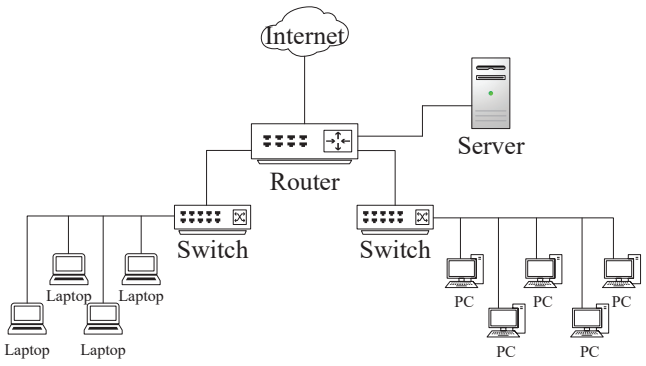


Fig. 5: Network Topology

To evaluate CoDAG's performance in a real environment, computing devices with different computing power are deployed, including a server, 5 Mac Minis and 4 Lenovo laptops, and they are connected in a LAN-based environment.

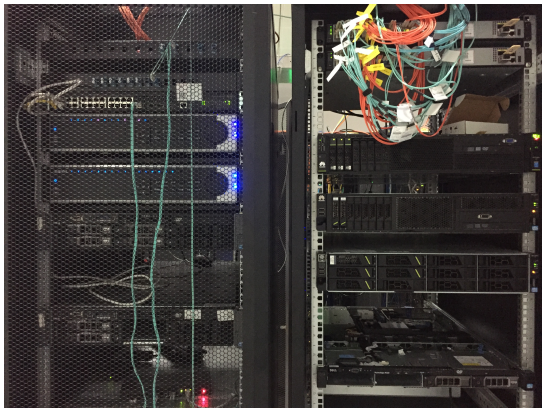


Fig. 6: Simulation Equipment

In a LAN-based environment, the link delay is very small, thus the information propagation delay of CoDAG will be negligible. To emulate a real-world network environment, we manually add some link propagation delay to the network. More specifically, we use the *NetEm* [41], which is a network emulation module to simulate the link delay in a LAN environment. In this paper, the average delay time is set to be 50ms [42]. In this case, the link latency will be added to the network, thus we can evaluate our CoDAG prototype in a real network environment.

Additionally, we design a script to produce enough transactions with random amounts and addresses. In CoDAG, every miner has a transaction pool to store the transactions that send to it. Initially, we let the trigger node run the script, and the new-generated transactions will be stored in the matched miner pools according to the transaction addresses, and wait to be bundled by the corresponding miners. In this case, miners will try their best to bundle transactions in their pools and append the block to the network.

In our experiments, we will evaluate the CoDAG's performance in different values of blocksize and width. The blocksize will scale from 10KB to 2MB, and the width will vary from 1 to 20. Meanwhile, we define *level formation time* as the required duration when a level is full of K blocks. Assume the level formation time of the system is T, then the average proposal time of every single block will be $\frac{T}{K}$. Therefore, the level formation time will directly influence the block proposal time in the network, and then influence the throughput of CoDAG. In this paper, we will study the relationship between the throughput and different values of level formation time. We will change the level formation time by altering the mining difficulty of the system. The more mining difficulty is, the more level formation time it will be.

Besides, we denote the *block utilization* as the proportion of valid blocks in the network. In our prototype, if the width of a level exceeds K, some blocks will become invalid and wait to re-append to the network again. Hence, we use block utilization to measure the percentage of valid blocks of the system, and investigate the relationship between block utilization and different values of blocksize and width.

The default simulation parameters are listed in Table.IV.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics

B. Experiment Results

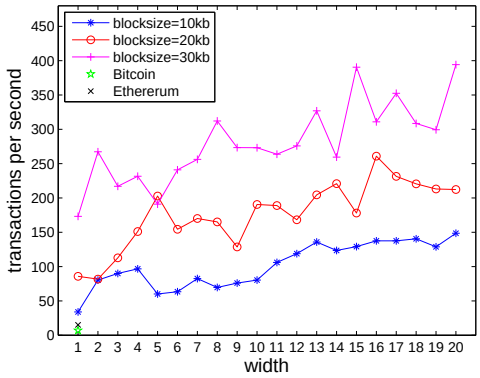


Fig. 7: Throughput of CoDAG

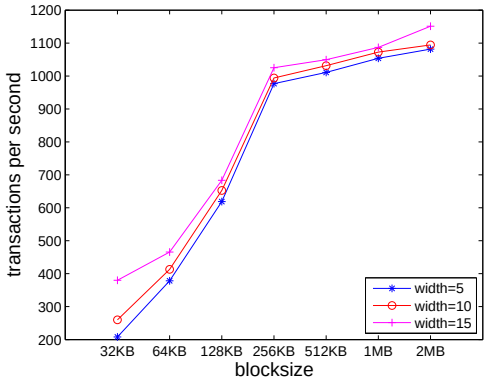


Fig. 8: Throughput of CoDAG

1) Throughput in Different Blocksizes and Widths: Fig.7 and Fig.8 present the CoDAG's throughput in different blocksizes and widths. In Fig.7, we can observe that given a fixed blocksize, as the width grows, the throughput of CoDAG shows an increasing trend. Meanwhile, given a fixed width, as the blocksize grows, the throughput will rise accordingly. For example, when the blocksize is fixed at 30KB, CoDAG with a width of 20 can achieve 394 tps, which is larger than other widths. Moreover, when the width is fixed at 20, CoDAG with a blocksize of 30KB is 2.6 times as blocksize=10KB and 1.9 times as blocksize=20KB respectively. Besides, CoDAG always performs better than Bitcoin and Ethererum. In Fig.7, CoDAG can achieve 394 tps, which is 56× Bitcoin's and 26× Ethererum's. Meanwhile, we also notice some throughput fluctuations as width of CoDAG grows. If blocksize is fixed at 30KB, when width increases from 13 to 14, throughput of CoDAG presents a slight decline. The reasons are twofolds: 1) a larger width will bring more block information for distributed nodes to synchronize and lead to longer propagation delay; 2) there is some instability of the transaction generation rate, and the amount of transactions may be instable, and one of our future works is to improve it.

Fig.8 also observes the relationship between throughput in different blocksizes and widths. We can see that CoDAG with larger values of width and blocksize outperforms the smaller one. Especially, when blocksize is 2MB and width is 15, CoDAG achieves the highest throughput of 1151 tps, which is 164× Bitcoin's and 77× Ethereum's. Moreover, we notice that the throughput of CoDAG soars as blocksize grows from 32KB to 256KB. However, when blocksize reaches a larger value, the throughput presents a slower growth. The reason is, due to the propagation latency, a block with larger blocksize will contain excessive transactions that can not be handled in time. Additionally, limited computing power of miners is also one of the restrictions.

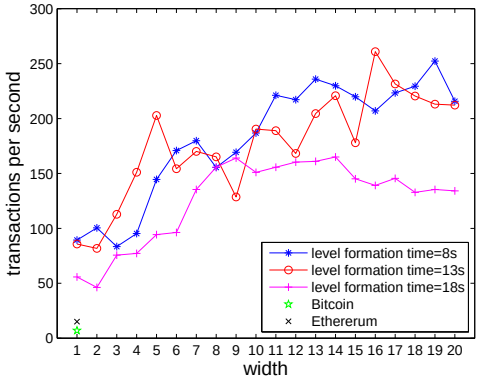


Fig. 9: Throughput of CoDAG in different level formation time when blocksize=20KB

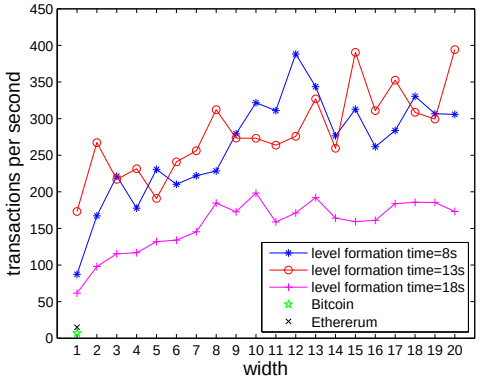


Fig. 10: Throughput of CoDAG in different level formation time when blocksize=30KB

2) Throughput in Different Level Formation Time: Fig.9 and Fig.10 depict the throughput in different level formation time. Generally, we can observe that CoDAG with a longer level formation time performs worse than smaller ones. In Fig.9, when blocksize is fixed at 20KB, CoDAG with 18*s* performs worse than 8*s* and 13*s*. Moreover, we notice no obvious deviation between 8*s* and 13*s*, which means when level formation time declines to a relatively small value, it will have limited effect on CoDAG's throughput. In theory, less level formation time will lead to faster block generation rate, and the throughput will rise accordingly. However, due to the capacity, excessive transactions and blocks can not be computed and verified by miners in time, thus the throughput gap is not obvious. When blocksize is fixed at 30KB, similar results can be obtained as Fig.10 depicts.

3) Block Utilization: Fig.11 provides the block utilization in different blocksizes and widths. Generally, the block

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2931157, IEEE Transactions on Industrial Informatics

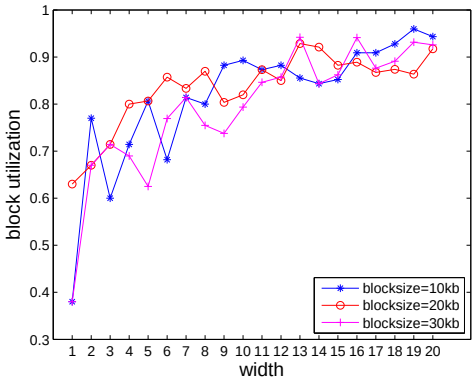


Fig. 11: Block Utilization of CoDAG

utilization will converge to 90% as width grows. This is because, if the width of CoDAG is too small, the new-generated block will not be aware that the target level has been filled with K blocks already. In this case, the new block will exceed the maximum value of width and be regarded an invalid block, thus the block utilization is low. As the width increases, the percentage of invalid blocks will decrease.

VIII. Conclusion and Future Work

In this paper, we devise an efficient and compacted DAG-based blockchain protocol CoDAG. Unlike the previous DAG-based solutions, blocks in CoDAG are organized in a DAG strcuture with a fixed width. We develop protocols and algorithms to maintain and secure the network, and transactions will be confirmed in a deterministic period. We present possible attack strategies in CoDAG and prove that it can resist these attacks. We also design a CoDAG-based IIoT architecture to achieve a better performance for the IIoT system. Finally, we implement a CoDAG prototype, and simulation results show that CoDAG outperforms the Bitcoin and Ethererum, which is suitable and efficient for IIoT.

In the future, to improve the CoDAG further, some modification schemes will be proposed, including: 1) tuning-up the parameters (e.g., blocksize and width), 2) combining with the hybrid consensus procotols.

References

[1] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] Q. Yan, W. Huang, X. Luo, Q. Gong, and F. R. Yu, "A multi-level ddos mitigation framework for the industrial internet of things," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 30–36, 2018.

[3] X. Sun and N. Ansari, "Dynamic resource caching in the iot application layer for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 606–613, 2017.

[4] N. Teslya and I. Ryabchikov, "Blockchain-based platform architecture for industrial iot," in *2017 21st Conference of Open Innovations Association (FRUCT)*, Nov 2017, pp. 321–329.

[5] Y. Xu, J. Ren, G. Wang, C. Zhang, J. Yang, and Y. Zhang, "A blockchain-based nonrepudiation network computing service scheme for industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3632–3641, June 2019.

[6] K. Huang, X. Zhang, Y. Mu, X. Wang, G. Yang, X. Du, F. Rezaeibagha, Q. Xia, and M. Guizani, "Building redactable consortium blockchain for industrial internet-of-things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3670–3679, June 2019.

[7] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3154–3164, 2017.

[8] X. Wu, B. Duan, Y. Yan, and Y. Zhong, "M2m blockchain: The case of demand side management of smart grid," in *2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2017, pp. 810–813.

[9] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, Sep. 2016, pp. 1–3.

[10] H. Liu, Y. Zhang, and T. Yang, "Blockchain-enabled security in electric vehicles cloud and edge computing," *IEEE Network*, vol. 32, no. 3, pp. 78–83, 2018.

[11] Z. Su, Y. Wang, Q. Xu, M. Fei, Y. Tian, and N. Zhang, "A secure charging scheme for electric vehicles with smart communities in energy blockchain," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4601–4613, June 2019.

[12] W. Liang, M. Tang, J. Long, X. Peng, J. Xu, and K. Li, "A secure fabric blockchain-based data transmission technique for industrial internet-of-things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3582–3592, June 2019.

[13] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

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

[15] L. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018, pp. 1545–1550.

[16] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, 2016, pp. 45–59.

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

[18] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.

[19] S. Popov, "The tangle," https://iota.org/IOTA_Whitepaper.pdf.

[20] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," *URL https://byteball. org/Byteball. pdf*, 2016.

[21] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: Serialization of proof-of-work events: Confirming transactions via recursive elections," https://eprint.iacr.org/2016/1159.pdf.

[22] Y. Sompolinsky and A. Zohar, "Phantom: A scalable blockdag protocol," https://eprint.iacr.org/2018/104.pdf.

[23] C. Li, P. Li, W. Xu, F. Long, and A. C.-c. Yao, "Scaling nakamoto consensus to thousands of transactions per second," *arXiv preprint arXiv:1805.03870*, 2018.

[24] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the internet of things: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1676–1717, Secondquarter 2019.

[25] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, 2016.

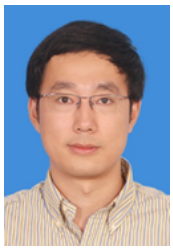
[26] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial internet of things," *IEEE transactions on industrial informatics*, vol. 14, no. 8, pp. 3690–3700, 2017.

[27] J. Wan, J. Li, M. Imran, and D. Li, "A blockchain-based solution for enhancing security and privacy in smart factory," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3652–3660, June 2019.

[28] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for iiot-enabled retail marketing atop pos blockchain," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3527–3537, June 2019.

[29] J. Huang, L. Kong, G. Chen, M. Wu, X. Liu, and P. Zeng, "Towards secure industrial iot: Blockchain system with credit-based consensus

mechanism," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3680–3689, June 2019.
[30] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.
[31] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.
[32] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*. ACM, 2018, p. 30.
[33] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Advances in Cryptology – EUROCRYPT 2017*, Cham, 2017, pp. 643–673.
[34] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 281–310.
[35] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016.
[36] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," 2014.
[37] R. Pass and E. Shi, "Hybrid consensus: Efficient consensus in the permissionless model," in *31st International Symposium on Distributed Computing (DISC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
[38] ——, "Thunderella: Blockchains with optimistic instant confirmation," in *Advances in Cryptology – EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds. Cham: Springer International Publishing, 2018, pp. 3–33.
[39] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 906–917.
[40] M. Rosenfeld, "Analysis of hashrate-based double spending," *arXiv preprint arXiv:1402.2009*, 2014.
[41] S. Hemminger *et al.*, "Network emulation with netem," in *Linux conf au*, 2005, pp. 18–23.
[42] L. T. Nguyen, R. Harris, and J. Jusak, "Analysis of networking and application layer derived metrics for web quality of experience," in *2012 IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 2012, pp. 321–325.



Laizhong Cui received the B.S. degree from Jilin University, Changchun, China, in 2007 and Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2012. He is currently an associate professor in the College of Computer Science and Software Engineering at Shenzhen University, China. He led the projects of the National Key Research and Development Program of China and the National Natural Science Foundation, and several projects of Guangdong Province and Shenzhen City. His research interests include future Internet architecture, edge computing, big data, IoT, computational intelligence, software-defined network and machine learning.

Shu Yang received his B.Sc. degree from Beijing University of Posts and Telecommunications and Ph.D. degree from Tsinghua University. His research interest includes network architecture and high performance router.

Ziteng Chen received his B.Sc. degree from Shenzhen University, Shenzhen, China, in 2018. He is pursing his M.Sc. degree in Shenzhen University. His research interest includes software-defined network and blockchain.

Yi Pan is currently a Regents' Professor and Chair of Computer Science at Georgia State University, USA. He has served as an Associate Dean and Chair of Biology Department during 2013-2017 and Chair of Computer Science during 2006-2013. Dr. Pan received his B.E. and M.E. degrees in computer engineering from Tsinghua University, China, in 1982 and 1984, respectively, and his Ph.D. degree in computer science from the University of Pittsburgh, USA, in 1991. His profile has been featured as a distinguished alumnus in both Tsinghua Alumni Newsletter and University of Pittsburgh CS Alumni Newsletter. Dr. Pan's research interests include parallel and cloud computing, big data, and bioinformatics. Dr. Pan has published more than 250 journal papers with over 90 papers published in various IEEE journals. In addition, he has published 194 papers in refereed conferences. He has also co-authored/co-edited 44 books. His work has been cited more than 10,000 times in Google Scholar and his current H-index is 54. Dr. Pan has served as an editor-in-chief or editorial board member for 20 journals including 7 IEEE Transactions. He is the recipient of many awards including an IEEE Transactions Best Paper Award, IEEE conference best paper awards, and several other conference and journal best paper awards, 4 IBM Faculty Awards, 2 JSPS Senior Invitation Fellowships, IEEE Outstanding Leadership Award, IEEE BIBE Outstanding Achievement Award, NSF Research Opportunity Award, and AFOSR Summer Faculty Research Fellowship. He has organized many international conferences and delivered keynote speeches at over 60 international conferences around the world.

Mingwei Xu received his B.Sc. degree and Ph.D. degree from Tsinghua University. He is a full professor with the Department of Computer Science. His research interest includes computer network architecture, high-speed router architecture. He is a member of the IEEE.

Ke Xu is with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He is also with the Beijing National Research Center for Information Science and Technology (BNRist), Beijing, China.