Blockcloud: Empowering IoT Through a Service-centric Blockchain

Zhongxing Ming*, Shu Yang†, Qi Li*, Dan Wang‡, Mingwei Xu*, Ke Xu*
* Dept. of Comp. Sci. & Tech., Tsinghua Univ., Tsinghua National Laboratory for Information Science and Technology
† Dept. of Comp. Sci. & Softw. Eng., Shenzhen University
‡ Dept. Computing, The Hong Kong Polytechnic University

***Abstract*—Connecting hundreds of billions of devices, the Internet of Things (IoT) has become a technology with large influence in people's life. However, the future of IoT is threatened by frail connectivity, poor scalability, absent trust, cracked security and broken business model. Researchers have proposed Service-centric Networking (SCN) to provide reliable connectivity and global scalability, while the trust, security and incentive of services in such approaches are hard to be guaranteed. Another direction is to use the blockchain technology to achieve a trust, secure and incentive IoT. However, most of those solutions are based on the assumption of a stable and well connected underlying network, which is often luxury in mobile and dynamic IoT scenarios.**

In this paper, we describe a new approach that combines the advantages of SCN and blockchain to empower IoT. We propose a service-centric blockchain architecture, which leverages a service-driven communication paradigm to support mobility and scalability. We design a new consensus mechanism and en efficient ledger structure to provide trust, security and incentive in a decentralized manner. Finally, we develop a service pricing and distribution mechanism that fairly protects the profits of both service providers and users of the network.

I. Introduction

With a predicted 20 billion devices by 2022 [1], the Internet of Things is increasingly a platform for accessing services that run anywhere. Applications such as smart healthcare, virtual and augmented reality, vehicle telematics, self-navigating cars/drones and multi-user ultra-high-definition telepresence are envisioned to be common in the near future (Table I). Unfortunately, the dream of a smart, safe and efficient future is being threatened. Inherited from the internet design, today's IoT architecture is facing great challenges by frail connectivity, poor scalability, absent trust, cracked security and broken business model [2]–[4].

A number of researchers propose a Service-centric Networking (SCN) to address the above problems [5]–[15]. SCN is a future internet technology that provides a service-aware network stack, where applications communicate directly on service names instead of addresses. In this way, SCN provides a clean-slate solution for interface failover, device mobility, and global scalability [15]. However, SCN is frail in terms of trust, security and economic incentive at the service level [16]. Another problem of SCN is the lack of compelling and sustainably profitable business models, which is at the same time, holding it back.

The research is supported by the National Natural Science Foundation of China under Grant 61625203 and 61572278, the National Key R&D Program of China under Grant 2016YFB0800102 and 2016YFC0901605, and the Chinese Postdoctoral Science Foundation under Grant 2016M591181.

TABLE I: IoT UNITS INSTALLED BASE BY CATEGORY (MILLIONS OF UNITS)

Category	2016	2017	**2018**	**2022**
Business: Cross-Industry	1,102.1	1,501.0	2,132.6	4,381.4
Consumer	3,963.0	5,244.3	7,036.3	12,863.0
Business: Vertical-Specific	1,316.6	1,635.4	2,027.7	3,171.0
Grand Total	**6,381.8**	**8,380.6**	**11,196.6**	**20,415.4**

Source: Gartner (January 2017)

Many studies use a blockchain to provide IoT with trust, secure and economically incentive properties [17]–[20]. The blockchain is the first technology that provides everyone with a working proof of a decentralized trust [21]. It also provides a frictionless Internet-native currency that allows for the first time to empower a sensor to take part in a world-wide service market and immediately earn profit, which strongly incentivizes the sensor to provide services to the network [22]. However, blockchain is an application-layer overlay whose functions rely on the underlying network. Unless the underlying network provides stable connectivity and sufficient scalability, the blockchain cannot play a positive role in real IoT scenarios, which are highly mobile and dynamic.

We present Blockcloud, a new approach that combines the advantages of SCN and blockchain to empower IoT. In Blockcloud, SCN forms the underlying physical network that provides connectivity, mobility and scalability. The blockchain logically operates on top of SCN to provide decentralized trust, security, fairness and economic incentive. The above design concept is shown in Figure 1.

Combining SCN and blockchain as an unified system and making it fit for IoT is a challenging task, with little previous work for reference. There are 5 aspects of our approach to address this challenge. (i) We design a layered structure to embed blockchain into SCN. (ii) We leverage a Service Access Layer (SAL) [15] to enable service-based communications. (iii) We design a *Proof-of-Service (PoS)* mechanism to reliably verify services. (iv) We develop a Compacted Directed Acyclic Graph (CoDAG) structure to effectively record transactions. (v) We devise a truthful continuous double auction (TCDA) mechanism to fairly distribute services.

We note that Blockcloud model is compatible with todays Internet and has a clear, simple evolutionary strategy. Like

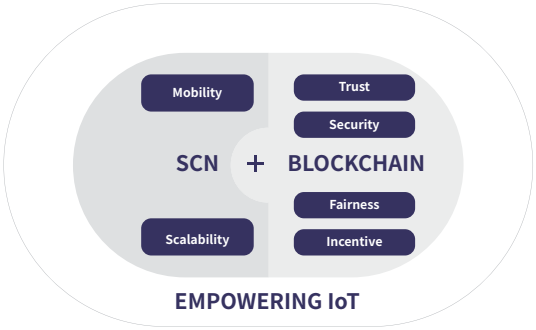

Fig. 1: Empowering IoT Through a Service-centric Blockchain

IP, Blockcloud is a "universal overlay": Blockcloud can run over anything, including IP. Thus Blockcloud's advantages in service distribution, application-friendly communication and naming, robust security, mobility and broadcast can be realized incrementally and relatively painlessly.

The contribution of this paper is as follows.

- We design Blockcloud, a new service-centric blockchain architecture to systematically address mobility, scalability, trust, security, fairness and incentive problems of IoT.
- We design *proof-of-service* consensus for reliable verification of IoT services.
- We develop CoDAG for fast recording of IoT transactions.
- We devise TCDA for fair pricing and matching of services.

The remainder of this paper is organized as follows. Section II presents the reason of leveraging a service-centric blockchain approach. Section III illustrates the architecture of the proposed approach. Section IV provides the detailed description of the service-centric communication paradigm. Section V shows the novel proof-of-service consensus for reliable service verification. Section VI presents the design of the Compacted DAG and prove its effectiveness. In Section VII, we describe the TCDA mechanism and prove that it is strategy-proof for both service providers and service users.

II. Background and Motivation

A. *Why a Service-centric Approach?*

IoT is exposed to more challenges than traditional internet. This is largely a result of the design of today's internet design, where communications are expected to happen between fixed entities [8]. However, in the IoT world, the scale and dynamic property of the connected sensors are facing great challenges in the sense of mobility, security, scalability, network management and business model [23], [24]. Unless executives make big strategic changes, they are set to be disappointed as they seek to scale todays IoT solutions to support tomorrows hundreds of billions of things [2].

Service-centric Networking (SCN) is a promising approach to evolve the Internet infrastructure away from a host-centric paradigm based on perpetual connectivity, to a network architecture in which the focal point is "named service". In this paradigm, connectivity may well be intermittent, end-point and in-network service can be capitalized upon transparently. The expected benefits are improved efficiency, better scalability, better mobility and better robustness in challenging communication scenarios [25]. For example, by placing an environment monitoring service near to the user, it may become possible for the user to access it locally when backhaul Internet connectivity is not available (e.g., after an earthquake).

The Blockcloud team proposes to improve IoT by the service-centric paradigm. This simple change to the hourglass model, allowing the thin waist to use service names instead of IP addresses for service achievement, makes service rather than its containers a first-class citizen in the IoT architecture. The service names in Blockcloud can be used to name a chunk of data in a conversation, as the TCP/IP transport signature plus sequence number does today, but they can also name a streaming service from YouTube directly, rather than forcing it to be embedded in a conversation between the consuming host and youtube.com.

Figure 2 shows three interconnected Autonomous Systems (ASes), each has one or more data centers acting as service execution environments. A service has been instantiated in two locations. From a service management and placement perspective, the orchestrator logic needs to decide in which service execution environment a service should be instantiated. Given this rich set of resources, SCN aims to optimize the location of individual service component instances according to the performance requirements of the application, the location of its users and according to the experienced demand.

*1) **Improving Mobility**:* In Blockcloud, the mobility of devices is inherently supported by adopting a service-based communication paradigm introduced by Serval [15]. The centerpiece of Serval architecture is a new Service Access Layer (SAL) that sits above an unmodified network layer, and enables IoT devices to communicate directly on service names. The SAL provides a clean service-level control/data plane split, enabling policy, control, and in-stack name-based routing that connects clients to services [15]. From the service subscriber's perspective, service is provided by the network and she does not need to manage the direct communication with the exact end-point who provides the service. In such a manner, end-points can seamlessly change network addresses, migrate flows across interfaces, or establish additional flows for efficient and uninterrupted service access (explained further in Section III).

To the best of our knowledge, Blockcloud is the first blockchain-based technology to comprehensively address the mobility problem of IoT.

*2) **Enhancing Scalability**:* In traditional IP network, most of the IoT platforms provide a centralized server to aggregate all IoT devices, data and services, publish them to the web portal, and manage the subscription membership. While such a centralized architecture ensures data/resource availability, it renders poor scalability and high bandwidth consumption due

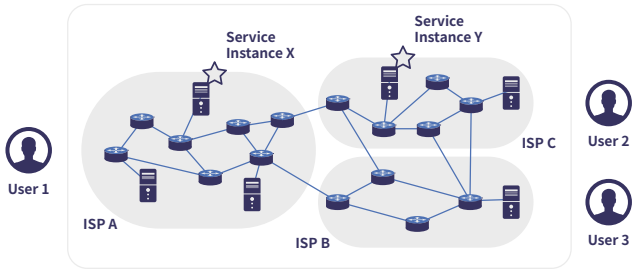

Fig. 2: Service-centric Networking Overview

to the high volume of control and data exchange. Scalability in the IoT spaces will be more challenging than web-scale or Internet-scale applications. The bisection bandwidth requirements for a centralized cloud solution are staggering, as the amount of data generated will easily exceed the reported trillion objects in Amazon S3 [26].

In Blockcloud, the distributed nature of services with peer-to-peer technology makes scalability possible. Service registration and discovery become the key features of Blockcloud, which can be easily achieved from the adoption of a Distributed Hash Table (DHT) network [27] (explained further in Section III).

B. *Why Blockchain?*

While the service-centric approach supports ubiquitous service mobility and good scalability, it does need some kind of centralized infrastructures for service registration, up-to-date service resolution and routing [15], [16]. Although these infrastructures can be provided by Internet Service Providers (ISPs) or using existing systems like DNS, the centralized operation is exposed to potential risks of malicious attacks and intrusive surveillance. Also, the published service cannot be credibly verified, which greatly harms the profits of end users when malicious services are subscribed.

Another serious problem is the incentive model, where service providers have no incentive to publish service to the network. This is because: 1) the quantity of profit for providing a service cannot be predicted; 2) the achievement of profit for providing a service cannot be guaranteed. The reason is that there is no means to build a fully trusted party (either centralized of distributed) to perform the global accounting and profit distribution in current SCN or traditional internet technologies.

The above problems can be well solved by a blockchain approach. One of the important contributions of blockchain is to enable the formation of a trust-generating mechanism in a decentralized manner, which makes the network a trust-generating machine. Another important contributors of blockchain is the decentralized governance and incentive mechanisms for ecosystem growth. Service providers get tokens with the help of smart contracts, with the certainty of getting profits protected by distributed consensus mechanisms.

By adopting a blockchain approach to enhance the service-centric network, we achieve the following gains.

1) ***Getting Trust****:* In Blockcloud, devices form a blockchain network where the system runs in a fully decentralized manner. We propose a novel Proof of Service (PoS) consensus mechanism to solve problems about guaranteeing truthful service. The facticity of services provided to the network is automatically checked and guaranteed by the PoS consensus. Cheating providers will be punished or even kicked out from the system. Successful decentralization of the IoT, however, will lie not just in being peer-to-peer, but also in being trustless: an environment in which there is no need for participants to be trusted and no centralized, single point of failure.

2) ***Protecting Security****:* In Blockcloud, the blockchain system and a peer-to-peer network known as Coral [27] form the secure backbone of the Blockcloud network. Communication operations between devices are embedded in the blockchain and service information is stored in Coral, routed by peer nodes (called service miners) and guaranteed by the *Proof-of-Service* consensus (explained future in Section III). The system therefore has no central point which is exposed to attackers and security is guaranteed by the consensus mechanism of distributed miners who are economically motivated to be honest.

3) ***Achieving Fairness****:* In Blockcloud, services are published/subscribed to/from the network. Service providers and subscribers are atomically matched by Blockcloud. The matching process is performed by a smart contract which cannot be controlled any end-point or a centralized party. The smart contract runs a Truthful Continuous Double Auction (TCDA) that prevents cheating and maximizes social welfare of the entire community. TCDA is mathematically strategy-proof, where there is no incentive for any of the traders to lie about or hide their private information from the other traders.

To the best of our knowledge, this is the blockchain-based technology to approach to the global fairness of service distribution in IoT.

4) ***Strengthening Incentive****:* By identifying and matching supply and demand for services in real-time, Blockcloud will create new marketplaces. These complex, real-time digital marketplaces will build upon the foundation established by IoT devices and blockchain networks to expand the reach of this transformation very quickly. They will enable new peer-to-peer economic models and foster sharing economies. In the end, the IoT is expected to make the physical world every bit as easy to search, utilize and engage with as the virtual world.

III. The Blockcloud Architecture

A. *Architecture Overview*

Blockcloud introduces new functionality on top of blockchains by defining a set of new operations that are otherwise not supported by the blockchain. Blockcloud has

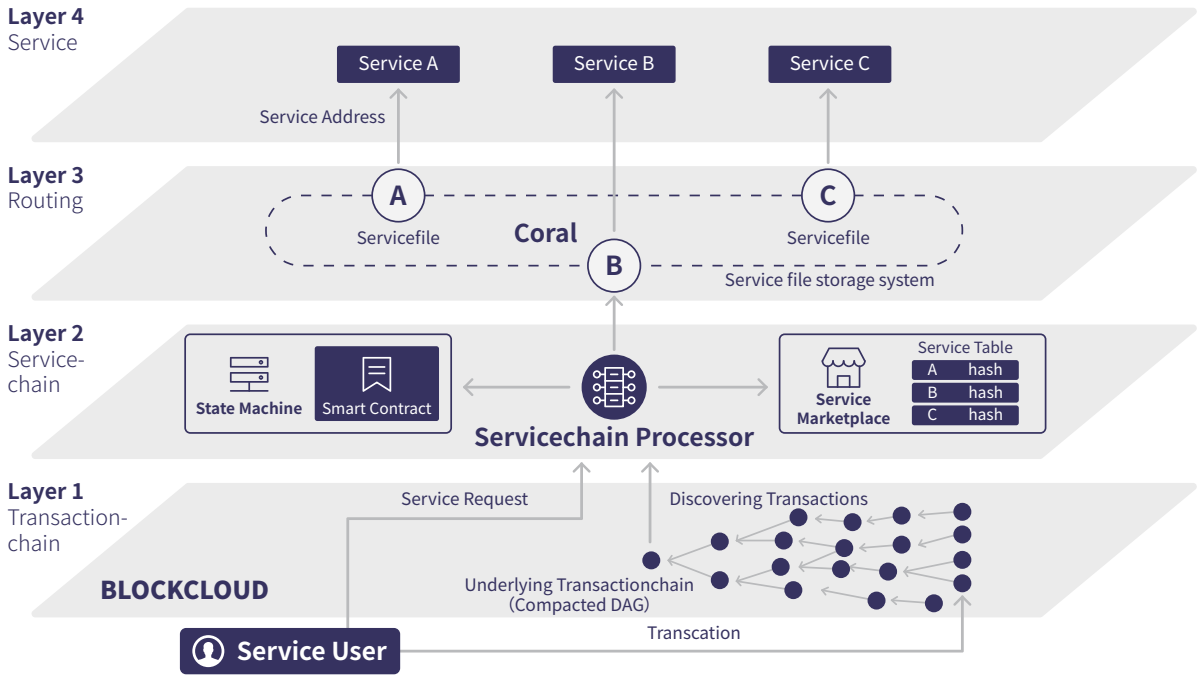

Fig. 3: The Blockcloud Architecture

four layers, with two layers (transactionchain layer and servicechain layer) in the control plane and two layers (routing layer and service layer) in the service plane.

B. *Blockcloud Layers*

1) ***Transactionchain Layer****:* The transactionchain occupies the lowest tier. Blockcloud operations are encoded in transactions on the underlying transactionchain. The transactionchain stores the transactions of the network as a global ledger and. While Blockcloud holds an evolving concept that provides the feasibility and compatibility of different underling blockchain technologies, due to the characteristic of IoT, the chosen technology for the transactionchain layer must be elastic, efficient, secure, scalable and cost-effective.

Currently, the Blockcloud team proposes to use Compacted Directed Acyclic Graph (CoDAG), which can be mathematically proved to be efficient for large-scale and dynamic IoT scenarios [28]. CoDAG improves the traditional DAG [29] structure to guarantee fast confirmation time and liveness property. In Section VI, we describe the design of CoDAG and prove its properties.

Agnostic of the Underlying Transactionchain: The design of Blockcloud does not put any limitations on which blockchain technology can be used with the Transactionchain, as long as the security and reliability properties are directly dependent on the underlying blockchain. We believe that the ability to migrate from one technology to another is an important design choice as it allows for the larger system to survive, even when the underlying blockchain is compromised.

2) ***Servicechain Layer****:* Above Layer 1 is a servicechain, which defines new operations without requiring changes to the underlying blockchain. Only Blockcloud nodes are aware of this layer and underlying blockchain nodes are agnostic to it. Blockcloud operations are defined in the servicechain layer and are encoded in valid blockchain transactions as additional metadata. Blockchain nodes do see the raw transactions, but the logic to process Blockcloud operations only exists at the servicechain level.

The rules for accepting or rejecting Blockcloud operations are also defined in the servicechain. Accepted operations are processed by the servicechain to construct a database that stores information on the global state of the system along with state changes at any given blockchain block. Servicechains can be used to build a variety of state machines. Currently, Blockcloud defines two state machines - a global service management system and a service matching and pricing system.

3) ***Routing Layer****:* Blockcloud separates the task of routing requests (i.e., how to discover service) from the actual providing of service. This avoids the need for the system to adopt any particular management service from the onset, and instead allows multiple service providers to coexist, including both commercial entity and peer-to-peer systems.

Blockcloud uses service files for storing routing information, which are similar to DNS zone files in their format. The servicechain binds names to respective hash (service file) and stores these bindings in the control plane, whereas the service files themselves are stored in the routing layer. Users do not need to trust the routing layer because the integrity of service

files can be verified by checking the hash(zone file) in the control plane.

*4) **Service Layer**:* The top-most layer is the service layer, which serves the actual services for the network. All service names are signed by the key of the respective owner of a service. By providing services outside of the transactionchain, Blockcloud allows arbitrary IoT services that a provided by a variety of IoT devices. Users do not need to trust the service provider because they can verify the authenticity of the service by *proof-of-service* in the control plane.

Blockcloud is designed to be a service oriented platform, whose design was motivated by the team's experience of years-long research and development of Service-centric Networking [15], [30], [31]. Communication in Blockcloud is driven by service names rather than addresses. A service name corresponds to a group of (possibly changing) processes offering the same service. Applications can use names to directly express their intent to publish or access specific services. This elevates services to first-class network entities (distinct from hosts or interfaces).

At the core of Blockcloud is a new Service Access Layer (SAL) that sits between the transport and network layers [15]. The SAL maps service names in packets to network addresses, based on rules in its service table managed by a blockchain (explained further in Section IV). Unlike traditional "service layers" which sit above the transport layer, the SAL's position below transport provides a programmable service-level data plane that can adopt diverse service discovery techniques. The SAL can be programmed through a user-space control plane, acting on service-level events triggered by active sockets (e.g., a service instance automatically registers on binding a socket). This gives network programmers hooks for ensuring service-resolution systems are up-to-date.

In Section IV, we provide details of the clean service-centric network stack.

C. Proof of Service

In practice, service publishers must convince their customers that they truly provide the service. In Blockcloud, we introduce a novel class of *proof-of-service* schemes that allows proving that a particular service is physically provided to the network as it is announced. In Section V, we describe the operations and prove the properties of *proof-of-service* in detail.

D. State Machines

Blockcloud supports smart contacts by running state machines in the servicechain layer. Blockcloud can construct an arbitrary state machine after processing information from the underlying transactionchain. A servicechain treats transactions from the underlying blockchain as inputs to the state machine and valid inputs trigger state changes. At any given time, the state machine can be in exactly one global state. Time moves forward as the underlying blockchain evolves and the global state is updated. A servicechain can introduce new types of state machines without requiring any changes from the underlying blockchain. Currently, Blockcloud introduces two state machines: 1) A service machine that represents the global state of the service management system, dealing with who provides a particular service. 2) A market machine that matches service publishers and subscribers, through a truthful continuously double auction algorithm that is explained further in Section VII.

E. Marketplace

Blockcloud is an incentive-driven marketplace for IoT Services. Anyone can publish a service to Blockcloud, and anyone can subscribe a service from Blockcloud. There is no central party to stop someone from doing so. Service providers who contribute to the network will be rewarded with tokens to form a positive, self-growing platform that continues to evolve on its own.

The Blockcloud Marketplace is a decentralized blockchain application built on Blockcloud platform itself. Service providers publish services to the Marketplace and users subscribe services from the Marketplace. It is also responsible for maintaining public feedback about the quality of services and the reputation of their providers, as well as other information. This information will help users evaluate reusable services.

A market mechanism for the efficient determination of service trades must address the following tasks [32]. First, the buyers and sellers who would benefit from trade must be identified. Second, these buyers and sellers must be matched so as to maximize total gains to trade. Third, a specific price must be determined for each matched buyer-seller pair.

In Blockcloud, we propose a novel truthful double auction mechanism, where multiple buyers and sellers compete with on another to buy and sell services. Continuous Double Auctions (CDAs) are one of the most common forms of marketplaces and have emerged as the dominant financial institution [33]. The CDA is used because it exploits the dynamics of the free market to balance demand and supply efficiently in a highly responsive and decentralized system. The Blockcloud team develops a truthful continuous double auction (TCDA) mechanism that allows users to bid for service. TCDA has the following properties:

- **Defending against price cheating**. Cheating traders can not increase the profit so that they do not have incentives to cheat.
- **Guaranteeing fairness**. TCDA treats sellers and buyers equally, where sellers and buyers have the same opportunity to earn profits.
- **Minimizing trading maintenance cost**. TCDA optimizes the trading maintenance cost to scale the system to support a large number of tradings simultaneously.

The preliminary work of TCDA was first used to support service trading in mobile networks, with the theoretical result published in [34]. In Blockcloud, we generalize the problem and use the core algorithm of [34] to build the service trading component of the Marketplace. In Section VII, we describe TCDA in detail.

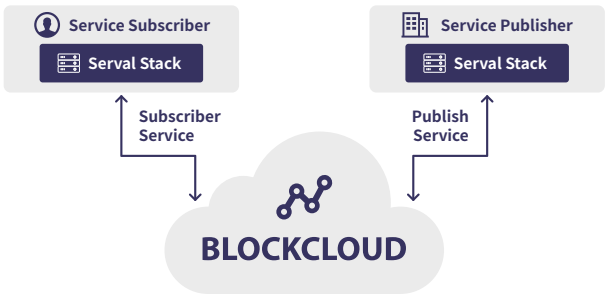

Fig. 4: Publishing and Subscribing Services in Blockcloud

IV. The Service-centric Network Stack

Researchers have conducted a number of studies on service-centric networking (or information-centric networking) since 10 years ago [7], [8], [10]–[15]. Among the SCN designs, Serval [15] provides a service-aware network stack, where applications communicate directly on service names instead of addresses and ports. The SAL provides a clean service-level control/data plane split, enabling policy, control, and in-stack name-based routing that connects clients to services via diverse discovery techniques.

By carefully investigating different SCN/ICN technologies and combining with previous work of the Blockcloud team in this research field [30], [31], we decide to use the Serval network stack to implement Blockcloud's service-based communication. Distributed applications (DAPPs) running the Serval stack can publish/subscribe services to/from Blockcloud, with the logic shown in Figure 4. In this paper, we briefly introduce the Serval network stack, with most of the descriptions coming from the Serval publication.

The Serval stack offers a clean service-level control/data plane split: the user-space service controller can manage service resolution based on policies, listen for service-related events, monitor service performance, and communicate with other controllers; the Service Access Layer (SAL) provides a service-level data plane responsible for connecting to services through forwarding over service tables. Once connected, the SAL maps the new flow to its socket in the flow table, ensuring incoming packets can be demultiplexed. Connectivity can be maintained across physical mobility and virtual migrations. DAPPs interact with the stack via name-based sockets that tie socket calls (e.g., bind and connect) directly to service-related events in the stack. These events cause updates to data-plane state and are also passed up to the control plane (which subsequently may use them to update resolution and registration systems). Figure 5 shows the above logic.

As such, Blockcloud gives service providers more control over service access, and clients more flexibility in resolving services. For instance, by forwarding the first packet of a connection based on service name, the SAL can defer binding a service until the packet reaches the part of the network with fine-grain, up-to-date information. This ensures more efficient load balancing and faster failover. The rest of the traffic flows directly between end-points according to network-layer forwarding. The SAL performs signaling between end-points to establish additional flows (over different interfaces or paths) and can migrate them over time. In doing so, the SAL provides a transport-agnostic solution for interface failover, device mobility, and virtual-machine migration.

To handle a wide range of services and deployment scenarios, the controller disseminates serviceID prefixes (hashed names) to Blockcloud, while the SAL applies rules to packets, sending them onward - if necessary, through service routers deeper in the network to a remote service instance. The SAL does not control which forwarding rules are in the service table, when they are installed, or how they propagate to other hosts. Instead, the local service controller (i) manages the state in the service table and (ii) potentially propagates it to other service controllers. The Blockcloud is responsible for registering, resolving and routing services, which supports different service deployment scenarios.

Due to page limit, we do not introduce more details about the Serval stack. To comprehensively understand the Serval architecture, we highly encourage the readers to read the Serval paper [15].

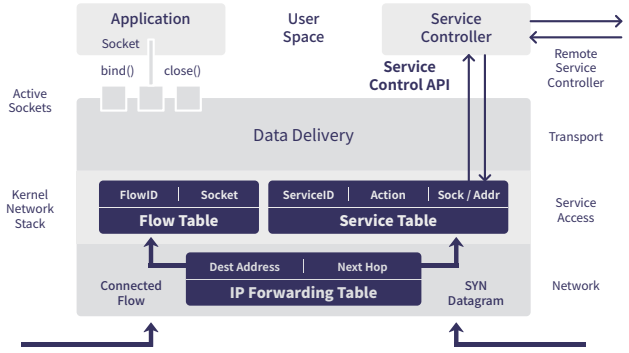

Fig. 5: Serval network stack

A. *Service Registration*

Blockcloud maintains a service naming system as a separate logical layer on top of the underlying transactionchain on which it operates. Blockcloud uses the underlying blockchain to achieve consensus on the state of this naming system and bind services to providers. Specifically, it uses the underlying transactionchain as a communication channel for announcing state changes, as any changes to the state of name-value pairs can only be announced in new blockchain blocks. Relying on the consensus protocol of the underlying transactionchain, Blockcloud can provide truthful manner for all operations supported by the naming system, like service registration, updating and matching.

Separation of the Control and Service Plane: Blockcloud decouples the security of name registration and name ownership

from the availability of data associated with names by separating the control and data planes. The control plane defines the protocol for registering services (identified by service names), creating (service, hash) bindings, and creating bindings to owning cryptographic keypairs, which is a logically separate layer on top.

The service plane is responsible for storing service information, mainly the service-hash pairs. It consists of (a) service files for discovering service by hash or URL, and (b) external storage systems for storing service information [27]. Service names are signed by the public keys of the respective service owners. Devices receive services from the service plane and verify their authenticity by checking that either the services hash is in the service file, or the service includes a signature with the name owners public key.

Our design not only significantly increases the data storage capacity of the system, but also allows each layer to evolve and improve independently of the other.

B. *Service Resolution*

As mentioned in §III-B3, services are resolve through alternative resolution networks, such as DHT-based ones. In such cases, a hash-based serviceID is forwarded through service tables, ultimately registering or resolving with a node responsible for the serviceID. This DHT-based resolution can coexist with an IANA-controlled resolution hierarchy, however, as both simply map to different rules in the same service table.

Currently, the Blockcloud team decide to use Coral [27] DHT. Coral is a peer-to-peer content distribution network that offers high performance and meets huge demand. It is completely decentralized and self-organizing, which enables people to publish content that they previously could not or would not because of distribution costs. It is also fit for mobile and dynamic networks, where a simple composition of previous solutions cannot achieve [27]. Due to space constraints, we omit most details of Coral from this paper.

We believe that the blockchain industry is still relatively young and evolving, and it is too early to pick a winning DHT technology. It is hard to predict which DHT technology will be operational and reliable five years from now. The layered structure of Blockcloud provides fully compatibility with new DHT technologies, which enables the system to be self-evolved.

C. *Service Routing*

The service files stored in Coral serves as pointers to the device who provides the service. The route to the service is returned by the servicechain to the subscriber. When the previous service provider moves or gets offline (can be quickly discovered by the Serval stack [15]), the route for the particular service is redirected to an alternative provider with the help of Serval. Meanwhile, the service file of the previous service provider is updated. In such a manner, Blockcloud supports ubiquitous mobility and service migration, which frequently happen in most real IoT scenarios.

The Coral network only stores service files if the service was previously announced in the transactionchain. This effectively whitelists the service that can be stored in the Coral. The key aspect relevant to the design of Blockcloud is that routes (irrespective of where they are fetched from) can be verified and therefore cannot be tampered with. Further, most production servers that can be used by service miners maintain a full copy of all service files since the size of service files is relatively small (roughly 4KB per file). Keeping a full copy of routing data introduces only a marginal storage cost on top of storing the blockchain data.

V. Proof of Service

To provide services to massive users, the whole network has to contribute resources, including storage, computation, bandwidth, routing and data. Evaluate the contributing resources is historically a tough problem [35]. In this document, we propose a flexible model to formalize the service, and a solution that incentivizes correct behaviors among all participant.

A. *Syntax Model*

For a single registered service, many network entities are related in the process, including routers, end servers, internal cache servers, various middle boxes. For each entity, the contributed resources are composed of storage, computation, bandwidth, routing and data, etc. We illustrate the resources as following,

- Storage: Storage of end users [36], servers, or even intermediate nodes can be used as disks or caches [37].
- Computation: According to the contract scripts, a device (or a set of devices) with enough capacity could be chosen to execute the scripts. The system designate a group leader, which will assign jobs to each node (group member), each node computes the intermediate results to the leader,
- Relay: The intermediate nodes that help deliver data to the destination, including the last-hop delivering devices, e.g., Wifi, 5G, VANET, and hop-by-hop delivering devices, e.g., WSN (Wireless Sensor Network), routers, VPN middle-boxes, etc. which will assemble the final results. For example, we can design a distributed firewall or intrusion detection system, where each node along the path can defend part of the attacking traffic [38].
- Bandwidth: Services may need to download or upload data to the network, a connection will be setup between devices storing the data and end users, data will flow along the connection. Bandwidth from different devices can be aggregated to promote the total bandwidth, especially in the multi-homing scenario [39].

We use e to denote an network entity, P_t to denote the total proved resources, and P_s^e, P_c^e, P_b^e, P_r^e, P_d^e to represent the proved storage, computation, bandwidth, relay, and data on entity e.

Thus, the total contributed resources can be seen as a combination of different resources on devices across networks.

$$P_t = \sum_e (P_s^e + P_c^e + P_b^e + P_r^e + P_d^e). \quad (1)$$

To prove the reality and contribution of services, several great challenges should be addressed, including, 1) the devices that act as a service provider should provide services correctly as promised, and the services could be verified by other devices; 2) the devices that provide services, and verify services should be properly incentivized and penalized, such that the whole network can provide "common good" services to users, avoid "verifier dilemma", including resource exhaustion DDoS attack and incorrect transaction attack [40].

Recently, many work have been devoted to prove contributions in a decentralized network. In [40], the authors proposed the concept of "consensus" computer, on which miners are expected to execute correctly. In [36], filecoin is used to reward those store data correctly, and penalize those do not. In [41], a new protocol called "verification game" is devised to verify the computation results and incentivizes correct behaviors. However, these work focus on storage and computation, which is not enough for a IoT network where many devices have limited storage and computation resources. While we draw lessons from previous work, we build a new protocol that can be extended to the IoT world.

To address the challenges, we define the following roles in the proof model.

Definition 1. *To provide service correctly, there involves three parties, service provider, subscriber and verifier.*

- *Subscriber ($\mathcal{S}$): who needs and subscribes a service.*
- *Provider ($\mathcal{P}$): who provide the service.*
- *Verifier ($\mathcal{V}$): who verifies the service.*

As shown in Equation 1, proof of service can be decomposed into proof of several basic elements, including storage, computation, bandwidth and relay. Among these elements, proof for storage and computation have been addressed, thus we focus on bandwidth and relay in this document. However, different with computation and storage, which return a deterministic results, bandwidth and relay are more probabilistic. We define *approximate QoS (Quality of Service)* to help prove relay and bandwidth.

Intuitively, the service provider promises its QoS when it registers the service initially. And verifiers will repeatedly try to check the service. We call a service is δ-approximate, if verifier try n times, and more than δ times success.

Definition 2. *The DHT table in the routing layer will maintain a approximate QoS record, which is* $\Delta = \{\Gamma = \{0,1\}^n, \{\mathcal{V}_i\}^n, counter\}$*, where* Γ *denotes current verification results table (0 represents success and 1 represents fail),* $counter$ *loops forth and back from 0 to* $n-1$ *indicating current position in the verification results table, and* $\{\mathcal{V}_i\}^n$ *denotes the verifiers set. We say* Δ *satisfies* δ*-approximate QoS if* $\sum_i i \in \Gamma \geq \delta$*.*

With the concept of δ-approximate service, a service provider will not be penalized once it fails to meet QoS requirements. Verifiers will try many times, and the state or results will be stored in the DHT table in a decentralized way, thus other verifiers can read the table before trying and write into it after trying. When the counter loop back (reset to 0), the system will decide whether reward or penalize the provider, and reward the verifiers. Finally, the specific incentive for rewarding can be set according to [40].

VI. Compacted Directed Acyclic Graph

A. A simple survey on related ledger structure

To achieve consensus, many ledger structures have appeared recently. They can be categorized into the following classes.

- **Permissioned protocol**: Achieving consensus through permissioned protocol is extensively studies in the past 30 years [42]. A new node needs to apply for permission before joining the network and transacting with other permitted nodes.
- **Permissionless protocol**: Different with permissioned protocol, nodes can freely join or leave the network in permissionless protocol, thus it is totally decentralized. It can be classified into on-chain model and off-chain model. In the off-chain model, part of the nodes can make transactions with each other privately, and summarize & upload all transactions to the whole network periodically (or eventually). While in the on-chain model, each transaction is directly submitted to the network.

  Off-chain model: To accelerate the transaction speed, many off-chain solutions have been proposed. Among them, lightning network [43] improves the scalability through setting up micropayment channels between users. Blockstream proposes sidechains [44], where independent sidechains could be devised with desired flexibility, and the sidechains can communicate with the main chain through a two-way peg. Other off-chain solutions rely on credit between transaction parties, or another trusted third parties, however, at the cost of sacrificing decentralization.

  On-chain model:

  1) **Linear Chain-based**: Bitcoin [45] is the first permissionless protocol that is widely deployed and every transaction could be carried out on-chain. Bitcoin uses "proof of work" as "key-less digital signatures" to prevent double-spending and guarantee safeness. Since bitcoin, many improved solutions were springing up, like GHOST [46], Bitcoin-NG [47], these solutions tried to improve bitcoin through different aspects, e.g., efficiency, fairness and usability. Among them, Ethereum [48] was considered to be most promising, it increases the transaction speed while supporting flexible smart contracts based on EVM (Ethereum Virtual Machine), on which users can develop decentralized application rather than crypto-currencies only.

     Although continuously being improved, the linear chain-based solutions are limited by their efficiency,

for example, bitcoin only supports 7 transactions per second and Ethereum supports 20 transactions per second, while VISA supports 24,000 transactions per second. In [49], it was proved that the speed is constrained by the maximum network delay to guarantee security.

DAG-based model: To break through the limit of linear chain-based solutions, an alternative structure based on Directed Acyclic Graph (DAG) was proposed [50]. Unlike linear chain-based solutions, which allows only one block to be attached at the end of the chain, DAG-based solutions allow multiple blocks to be attached simultaneously. Many current projects are using DAG-based ledger structure, such as IOTA [51] and Byteball [52]. Although IOTA is famous due to its better scalability and lower transaction fee, it has also been widely criticized for its non-deterministic confirmation time and huge flooding traffic. Byteball maitains a main chain, which is selected by witnesses to establish total order between blocks. The witnesses are reputable users that could be elected by other users. Byteball makes a tradeoff between deterministic criteria and decentralization.

In the academic area, PHANTOM [53] and SPECTRE [54] were proposed, and their performances were proved more strictly. SPECTRE enjoys high throughput and fast confirmation, but it can not be extended to support full linear ordering over all blocks. PHANTOM keeps full ordering, but at the cost of slow confirmation process.

- **Hybrid Consensus:** Hybrid consensus protocol [55] combines the advantage of both permissioned and permissionless protocol, and tries to achieve the best of both worlds. Nakamoto himself first proposed in a forum that bitcoin can be used as a backbone protocol for solving Byzantine agreement [56]. In a project called Thunderella [57], hybrid consensus is used to support instant transactions.

Above all, the permissioned protocol is faster, but works in a centralized manner. Permissionless protocol allows miners to join or leave freely, but sacrifices efficiency and works much slowly. Although off-chain solutions can improve it, these patch-like solutions may cause new problems. Hybrid consensus protocol combines the benefits of both worlds, but it still relies on permissionless protocol as the backbone protocol for solving Byzantine agreement. In this document, we adopt the hybrid consensus while improving the underlying backbone protocol, for the IoT world with much faster transaction speeds.

B. CoDAG Framework

In the IoT network, IoT devices are usually constrained by battery, computing power, communication bandwidth, etc. To solve the problem, we design a framework shown in Figure 6. In the framework, besides the *miners* and *devices*, there is an additional role called *gateway*, which ubiquitously exists in the IoT world.

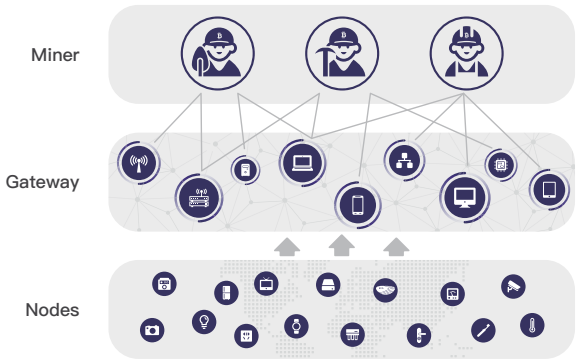

Fig. 6: CoDAG Ledger Framework

- Miner: It is responsible for maintaining the ledger, which represents the consensus over the network, through an improved DAG structure that will be illustrated in the following section. Like [55], miners have slow and fast modes: 1)slow mode using a DAG-based ledger structure, slow mode is also responsible for electing members for fast mode; 2) fast mode use Byzantine agreement, where the members are elected by slow mode in a decentralized way, rather than a centralized way [58].
- Gateway: It could be a device with enough resources to run the protocol, like a cellphone connected by various wearable devices, or a smart car connected by many in-car devices, or a 5G base station connected by numerous IoT devices. The gateway is responsible for downloading part of the ledger (most recently), maintaining an encrypted communication channel with the devices, and making transactions with other gateways. The gateway is like an important intermediate node in the off-chain solutions [43]. Because the gateway is usually a reliable and trusted entity in the IoT world,
- Devices: It only needs to upload its payment and other data to the gateway, and receive the results. If the device has enough capacity, itself can also be the gateway.

C. CoDAG - A Compacted DAG-based Ledger structure

The IoT network has numerous devices and resources, thus high frequency transactions between devices should be supported. Initially, the miners enter into the slow mode, which uses a permission-less protocol to achieve consensus and elect members for the fast mode. The members are elected in a decentralized way, i.e., the most recent M miners that have found new blocks being confirmed by the ledger protocol. After election, the miners enter into fast mode using Byzantine agreement. when members misbehave or encounter attacks, the system fall back to slow mode.

Although hybrid consensus [57] greatly improves the transaction speeds of permission-less protocol, and works in a decentralized way. However, its underlying protocol influences the period that it takes to enter into fast mode. More worse,

if under attack, the speeds will be lower by orders of magnitude, as it fall back to slow mode. In this document, we use DAG, which is suitable for asynchronous operations in decentralized environments, as the ledger structure to improve the permission-less on-chain protocol.

Besides, traffic congestion, uncertain worst-case confirmation time brought new challenges to these solutions. In this document, we design a new ledger structure called CoDAG (**Co**mpacted **D**irected **A**cyclic **G**raph) structure. Transactions are enveloped in blocks, which is organized as directed acyclic graph (DAG). The DAG is a gossip-like structure, that fits well in asynchronous operations. Unlike previous DAG-based structures, the new graph is well connected between the honest blocks, and has a limited width.

TABLE II: Summary of Notations

G	Graph or DAG in this document
V	set of nodes
E	set of directed egdes
l_v	level of node v
w_l	width of level l
K	maximum width
C_v	connectivity of node v
R_v	reverse connectivity of node v
$\mathcal{C}_l$	candidate set of level l

In the new ledger, let $G = (V, E)$ be a DAG, where V is the set of nodes or blocks (in the following text, we use node and block interchangeably), and E is the set of directed edges. In the graph, we use l_v to denote the level of a node in the graph, which is the length of the longest path from the genesis node to the current node. For example, in Fig. 7, the level of node 8 is 3. Each level has a width, which is the number of nodes in the level. For example, there are 3 nodes in level 3, and we use w_l to denote the width of level l. For a DAG, we call its width is constrained by a constant K, if $\forall l$, $w_l < K$, e.g., the DAG in Fig. 7 is constrained by 3. The width represents the maximum number of blocks the system can generate during a round. The width could also be adjusted (or self-adaptive) according to the transaction generation rate in the network. In the worst case, the structure degenerates into Bitcoint's linear blockchain structure if the width equals to 1. The width also influences the puzzle difficulty to guarantee the security level of the whole system.

In the graph, each node points to K nodes in the previous level, we call the number of paths from a node to the genesis *connectivity* of the node, e.g., the connectivity of node 8 is 9, including 9 paths:

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

Let C_v denote the connectivity of node v. We call the leaf nodes *tips*, and the tip that has the largest connectivity number is called the *navigator* of the graph. If there exist multiple tips with equal connectivity, then any one of them could be be the navigator. For example, node 13, 14, 15 have the same connectivity, and node 14 is randomly selected as the navigator. We call the number of paths from the navigator to the a node *reverse connectivity* of the node, and use R_v to express it. For example, if node 14 is the navigator, the reverse connectivity of node 8 is 3, including 3 paths:

$\langle 14\ 10\ 8\rangle$, $\langle 14\ 11\ 8\rangle$, $\langle 14\ 12\ 8\rangle$.

The basic idea of our protocol is that the honest nodes should be well connected (with higher probability), and has higher reverse connectivity for the current navigator.

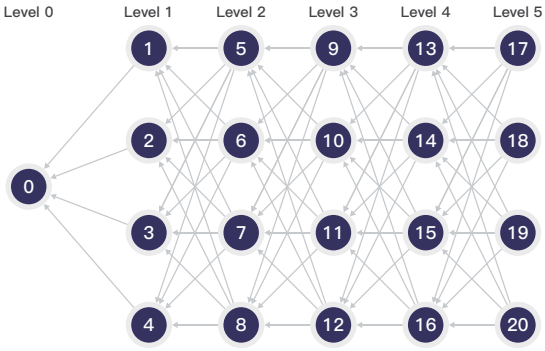

Fig. 7: A simple example for CoDAG

D. Protocol on CoDAG Overview

To illustrate our new protocol, we first recall the Nakamoto's protocol.

Nakamoto's protocol in a nutshell: To defend against "sybil" attack, in Nakamoto's protocol, miners try their best to mine a block, more specifically, solve a time-consuming computational puzzle that depends on the history transactions. The block is appended to a chain of blocks, and confirmed to be stable if 6 more blocks are appended after the block. Due to network delay, forks may appear when a miner computes a block while another new block is about to arrive. All honest miners choose the longest chain to append new blocks when forks exist.

The Nakamoto's protocol satisfies two important properties [59], 1) **consistency**: all honest miners in the network have a consistent view of the ledger; 2) **liveness**: if a transaction is successfully submitted, it will appear in the ledger of all honest miners.

Protocol on CoDAG: Like Nakmoto's protocol, each miner will computer blocks through solving puzzles, despite its much faster generation rate in CoDAG. Each miner generates blocks through solving puzzles, each newly generated block is appended to the CoDAG as a tip, and validates two blocks in the previous level, i.e., points to the two blocks in the graph.

For faster convergence speed, we set an upper limit K on the width of the graph. However, due to network delay or adversarial nodes, there may exist forks, i.e., more than K nodes in some level. Under such circumstance, nodes with higher reverse connectivity have higher priority compared with those with lower reverse connectivity. The top K priority nodes in the same level are considered to be *candidates* for this level. We use $\mathcal{C}_l$ to denote the candidate set of level l. A candidate is *stable* at some moment, if the probability of the

event where the node fails to be elected as a candidate in the future is negligible. The miner that generates a stable block will be rewarded by the system, like Nakamoto's protocol. A special case is that several nodes with the same priority belong to the borderline candidates, in such case, all these nodes will be classified as candidates and share the rewards.

In the protocol, the width should be set properly. If the DAG is too narrow, the DAG node generation speed will be slow; if the DAG is too wide, the confirmation time will be very long.

E. CoDAG Block Generation and Candidate Selection Algorithm

Block Generation Algorithm: We use K to denote the maximum level of the graph G, and $\langle v, w\rangle$ to denote a directed edge that points from v to w. We design algorithm $Generate()$ to compute the blocks the places where a new generated block x is to be appended. Algorithm $Generate()$ first computes the connectivity number of each node, and points to two nodes that have the largest connectivity in the previous level. In $Generate()$, we use algorithm $Connectivity()$ as an intermediate function, which takes a DAG and a node as the input, and computes the connectivity from all nodes (behind it) to it.

The input of the algorithm is the current DAG G and the new block x, the output is a new DAG with x and two directed edges that points from x to two previous nodes. Let L be the maximum level of graph G.

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      y, z ← two nodes that have the largest connectivity
         in level L − 1
6      V ← V ⋃{x}
7      E ← E ⋃{⟨x, y⟩, ⟨x, z⟩}
```

Theorem 1. *The computation complexity of algorithm $Generate()$ is $O(L \times K^2)$.*

Proof. Algorithm $Connectivity()$ is based on dynamic programming, line 5 in algorithm $Connectivity()$ loops for L times, line 6 loops for K times, line 7 loops for K times. Thus the complexity of the algorithm $Generate()$ is $O(L \times K^2)$. □

Candidate Selection Algorithm: We design algorithm $Select()$ to select the candidate for some level in the DAG. $Select()$ computes the candidate node for a layer in the DAG. It first computes the connectivity of each node, and the node with the largest connectivity is set to be the navigator. Secondly, for a specific layer l, $Select()$ computes the reverse connectivity for each node in the layers. At last, the nodes that have top K largest reverse connectivity are selected as the candidates. The input of the algorithm is the DAG and a specified layer, the output is the candidate set.

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      x ← select an element from N
7      R_v ← 0, ∀v ∈ V, R_x = 1
8      for L − 1 ≥ k ≥ l do
9          for ∀v ∈ {v ∈ V|l_v = k} do
10             for ∀w ∈ {l_w = k + 1} do
11                 if ⟨w, v⟩ ∈ E then
12                     R_v+ = R_w
13     T ← |{v|l_v = l}| ≤ K 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]}}
```

Theorem 2. *The computation complexity of algorithm $Select()$ is $O(L \times K^2)$.*

Proof. The complexity of algorithm $Connectivity()$ is $O(L \times K^2)$ as previously proved, the complexity of line 8-12 is $O((L - l) \times K^2)$, thus the total complexity is dominated by $O(L \times K^2)$. □

F. Well Connected Channel

The CoDAG structure is different from chain-like structure, as it adopts a flexible structure that is more suitable for asynchronous operations. It is also different with previous DAG-based structure, it adopts a more compacted structure, where nodes are more well connected. Intuitively, CoDAG is more like a well connected channel, and the width of the channel can be adjusted according to user demands.

We consider the scenario where an adversary tries to generate an alternate channel with higher connectivity than the honest channel, and takes control of the navigator node. Like bitcoin whitepaper [45], we define the following notations:

- p = the proportion of computing power that the honest party controls
- q = the proportion of computing power that the adversary party controls
- q_z = probability that the adversary will ever catch up from z levels behind

Lemma 1.

$$q_z = \begin{cases} 1, & \textit{if } p \leq q. \\ (q/p)^z, & \textit{otherwise}. \end{cases} \tag{2}$$

Basically, to control the navigator, the adversary needs to produce more nodes in the alternate channel than the channel produced by the honest miners. In each level, if both parties start at the same time, the probability that the honest party first finds the next w block is p, the probability that the adversary party first finds the next w block is q. Thus, the left proof is the same as [45].

Next, we consider how long a transaction will be confirmed. A node in some level selected as the candidate could be taken over by another node in the same level.

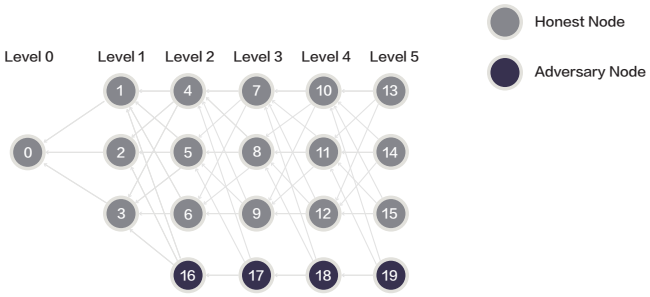

Fig. 8: Adversary starts an alternate channel

Lemma 2. *If the reverse connectivity of a node v locates z level away from the largest level, then the possibility that its candidate identity could be overtaken is:* $1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!}(1 - (q/p)^{z-k})$

Proof. Suppose there are two nodes in level l, node a produced by honest miners and b produce by adversary miners, suppose $R_a > R_b$ when the largest level of the graph L satisfies $L - l \leq z$, then the probability $Pr(R_a > R_b)$ in the future equals $1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!}(1 - (q/p)^{z-k})$.

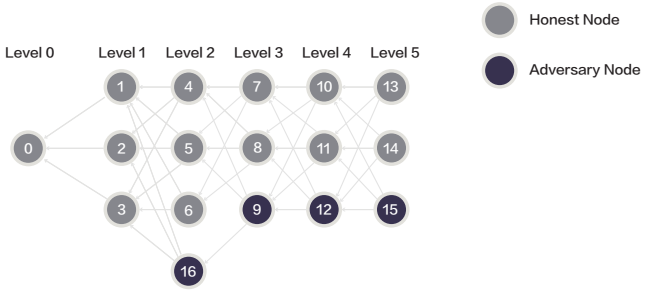

Fig. 9: Adversary competes in the same channel

There are two strategies in the channel-based scenario. The first is the adversary will start an alternate channel the level, as shown in Figure 8. In this case, the best strategy for the adversary party is to develop a complete separate channel, i.e., adversary node will only point to adversary node, but not honest node in the previous level. The model then is the same as bitcoin.

Different with bitcoin, there is a second scenario, where the adversary competes in the same channel with the honest miners, that is, the honest nodes may point to the adversary node, as shown in Figure 9. Obviously, the best strategy for the adversary is to point to adversary nodes at first, and point to the honest nodes with its left pointers. For the honest nodes, it will first point to honest nodes, and point to the adversary nodes with its left pointers.

We first prove that in any level $k > l$, for any 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 any 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 $l+1$, 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 $C_{w^{k+1},b} = Y' * Q + Y * (K - Q)$.

Secondly, we prove the above lemma. For a level $k > l$, let $\mathcal{R}(k) = \sum_{\forall v^k} R_{v^k}$ be the total reverse connectivity of all honest nodes in level k, and let $\hat{\mathcal{R}}(k) = \sum_{\forall v^k} R_{w^k}$ be the total reverse connectivity of all adversary nodes. Suppose $\frac{\hat{\mathcal{R}}(k)}{\mathcal{R}(k)} \leq \epsilon$ should be satisfied to make the adversary party catch up with the honest party. In level $k+1$, $\mathcal{R}(k) =$

$\mathcal{R}(k+1)*P+\hat{\mathcal{R}}(k+1)*(K-Q)$ 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 $\frac{Q(K-P)}{P(K-Q)} \leq 1 \leq x^2$ with higher probability, because the honest party own more computing power. Thus, $\frac{\hat{\mathcal{R}}(k+1)}{\mathcal{R}(k+1)} \geq \frac{P\epsilon}{Q}$ is satisfied with higher probability. Thus, the second strategy will fall back to the first strategy. □

For the second strategy, intuitively, each node is considered to have K votes, and the votes are given to the nodes in the previous level. For the honest nodes, the votes are first given to the honest nodes, the left are randomly given to the adversary nodes in the previous level; while for the adversary nodes, the votes are first given to the adversary nodes, the left are randomly given to the honest nodes in the previous level. However, if the honest party owns larger portion of the computing power, then most votes will be given to the honest nodes, and the votes will be accumulated with the level growing.

VII. Truthful Continuous double Auction

A. Preliminaries

In this paper, we consider buyers and sellers of services have multi-unit capacities and differentiated revenues and costs for auctions and their private information cannot observed directly by traders. Moreover, each trader bids in a manner that permits the exploitation of profit opportunities arising from the pricing behavior of other traders. To summarize, we base the auction on the independent, private value model.

Definition 3: The **independent, private value model** is the model in which each trader knows how much he/she values the item, he/she does not know how much others value it, and his/her value for it does not depend on the values of others.

We require the trading system to be strategy-proof so that traders have no incentive to lie about their price.

Definition 4: An auction strategy is said to be strategy-proof if there is no incentive for any of the traders to lie about or hide their private information from the other traders.

The strategy-proof property has been used in a wide range of computer network problems [60]–[62]. A typical application of strategy-proof is Vickrey-Clarke-Groves (VCG) mechanism [63]–[65], in which in order to maximize the individual welfare, a trader should seek to maximize the social welfare of the public system. While the VCG mechanism achieves a socially optimal allocation in an incentive-compatible manner, it is not suitable here as it puts high computational costs to the central organizer.

To balance supply and demand, we need to design a system with a competitive equilibrium.

Definition 5: A competitive equilibrium in a market for a positively valued good is a (positive) unit price P, a total quantity supplied $Q_S(P)$, and a total quantity demanded $Q_D(P)$ such that $Q_S(P)=Q_D(P)$. That is, the total quantity supplied must equal the total quantity demanded.

The notation $Q_S(P)$ and $Q_D(P)$ indicates that these supply and demand quantities depend on the price P. The total supply at each given price is simply the sum of the quantities of good that each seller plans to sell at that price. Let $q_i(P)$ denote the quantity of the service seller i plans to sell at price P, then $Q_S(P)$ takes the following form.

$$Q_S(P)=\sum_i q_i(P). \tag{3}$$

The competitive profit $W_i(P)$ of seller i takes the following form:

$$W_i(P)=Pq_i-p_iq_i. \tag{4}$$

The marginal price parameter p_i denotes the price that the seller expects to sell the service.

Similarly, the total demand at each bid price is the sum of the amount of commodities at that price. Let $q_j(P)$ denote the quantity of the commodity buyer j wants to buy at price P

$$Q_D(P)=\sum_j q_j(P). \tag{5}$$

The competitive profit $W_j(P)$ of buyer j takes the following form:

$$W_j(P)=Pq_j-p_jq_j. \tag{6}$$

The marginal price parameter p_j denotes the price that the buyer expects to buy the service.

Note that a competitive equilibrium is achieved at any price P where $Q_S(P)$=$Q_D(P)$.

Definition 6: An **equilibrium price** is the price at which quantity supplied is equal to quantity demanded.

It is possible that infinitely many competitive equilibria and market clearing prices exist. A competitive equilibrium ensures that demand meets supply in a free market populated by rational and selfish agents [66]. The concept is widely used in various economic areas, such as the New York Stock Exchange, the NASDAQ Stock Market [67], and B2B e-commerce websites (e.g., Free Market). In our mechanism, we determine a uniform equilibrium price in each auction round. We then use the equilibrium price to determine the winners of the auction. The marginally matched buyers and sellers use the uniform equilibrium price for all units. In such a way, both buyers and sellers achieve the same privilege so that the system is fair to both buyers and sellers.

In contrast to traditional auction markets such as electricity and spectrum, the Blockcloud Marketplace may involve tens of millions of traders. As a result, the system-level cost of maintaining the trading may be significantly high. Generally speaking, the more transactions the system manages, the higher trading cost it takes. Therefore, existing auction algorithms cannot be applied to TCDA as they cannot control the number of transactions. In this paper, we reduce the trading cost by reducing the number of transactions. Next, we will propose a three-stage mechanism to solve the problem. We

build a linear programming model targeted at minimizing the number of transactions and develop an algorithm to solve the problem that allows the matching result of buyers and sellers.

B. Algorithm overview

Figure 10 shows the TCDA framework that is based on a multilateral and continuous double auction. TCDA consists of traders who provide or subscribe service and a Marketplace that manages the auction. The prices for services are specified for each buyer and seller. The auction is anonymous and these parameter values are private to each trader.

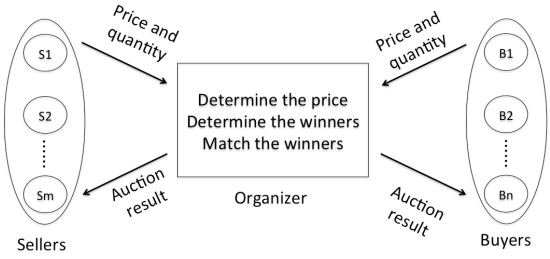

Fig. 10: The TCDA framework

The auction is performed in rounds. At the beginning of each auction round, traders submit price and quantity offers to the Marketplace. Each bid and ask consists of a single price-quantity pair. The price can be submitted via an API that is provided by Blockcloud. At the end of each auction round, the Marketplace matches the price and quantity offers received during the round in accordance with publicly known protocols, and reports these matches back to the buyers and sellers. The prices set by the Marketplace in each auction round is uniform (set equal across all matched buyer-seller pairs). Upon receiving the auction result, the matched buyers and sellers carry out their assigned trades and get their profit outcomes. The system is a continuous double auction. In the real IoT scenario, traders may continuously join and leave the system. The traders that are not matched in the previous round and the new participants can bid in next rounds.

C. A Three-stage Double Auction Mechanism

The auction in TCDA is performed under the following three procedures sequentially. (i) Determining the transaction price, (ii) determining the winners of the auction, and (iii) determining the quantity level for each matched buyer-seller pair. We present three algorithms that handle the issues above. Figure 11 shows the TCDA that achieves the equilibrium price. The winners of the auction and the matched buyer-seller pairs are determined in a sequential manner. The system is strategy-proof and fair to both sellers and buyers, and can minimize the trading cost in terms of the number of transactions.

1) Procedure 1 - Determining the transaction price: In this section, we calculate the market-clearing price based on which the winners of the auction are determined. Let m be the number of sellers in the system, M be the total units of service under sell (for short, supply), n be the number of buyers in the system, and N be the total demanded service wanted by the buyers (for short, demand). Let l denote the number of total traders, i.e., $l = m + n$. In an auction round, each trader reports K, i.e., the units of service supplied (or demanded), and the estimated price p to the organizer. We can have the following definition.

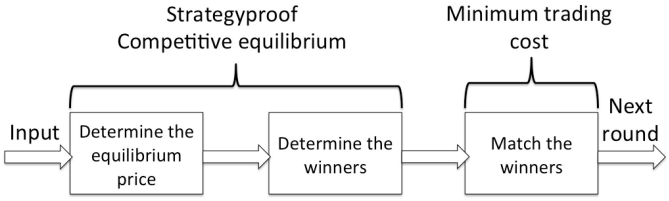

Fig. 11: The trading procedures in TCDA.

$$p = \begin{cases} < 0, & \text{if the trader is a buyer,} \\ > 0, & \text{if the trader is a seller.} \end{cases} \tag{7}$$

To calculate the market-clearing price, we sort the absolute value of the reported prices in a descending order. Let p_i denote the ith highest absolute value, K_i denotes the supply (demand) corresponding to the trader with price p_i, then the price set P is defined as follows:

$$\{\underbrace{p_1, p_1, \cdots, p_1}_{K_1}, \cdots, \underbrace{p_i, p_i, \cdots, p_i}_{K_i}, \cdots, \underbrace{p_l, p_l, \cdots, p_l}_{K_l}\}, \tag{8}$$

where $|p_1| \geq |p_2| \geq \cdots \geq |p_l|$.

Each listed price represents the price for each unit of supplied (or demanded) service. The benefit of constructing such a price set is that the organizer can change the auction with l traders to an auction with $L = M + N$ traders and each trader supplies or demands a unit quantity of service. Given the above price set, we define the price located at the $(M{+}1)$-th position of the set as the *$(M{+}1)$-th price*. Algorithm 4 presents the pseudo-code of finding the $(M{+}1)$-th price.

Note that, the upper bound of the absolute value of p_i is the carrier price, which is a definite number. In real practice, for a practical online service trading system, the accuracy of the price is expected to be definite. Therefore, if we use bucket sort, $p[1...L]$ can be sorted with a computational complexity $O(L)$. As a result, the computational complexity of Algorithm 4 is $O(L)$ as well.

Algorithm 4: Price Determining Procedure (the set of price, i.e., p[1...L], and M supplied service)

```
for (i=1; i ≤ M; i++) do
    p[i] = |p[i]|;
end for
sort p[1...L] in a descending order
return p[M+1];
```

Consider the value of the $(M{+}1)$-th price is p_s. It means that the $(M{+}1)$-th price is the s-th highest price among all the prices.

2) Procedure 2 - Determining the winners: Given the market clearing price, we then present the market clearing rule that determines the winners. We call the set of traders who win the auction at the end of the auction round the *trading set*. We prove that the auction achieves a competitive equilibrium under this rule.

Market Clearing Rule.

- If the trader at the (M+1)-th position is a seller, then the trading set consists the following: (i) All the buyer prices that are before the (M+1)-th position. (ii) All the seller prices that are after the (M+1)-th position, and (iii) The price at the (M+1)-th position.
- If the trader at the (M+1)-th position is a buyer, then the trading set is the same as the above trading set except that the price at the (M+1)-th position is not included.

Algorithm 5 shows the pseudo-code of determining winners. The computational complexity of Algorithm 5 is $O(L)$.

Algorithm 5: Market Clearing Procedure (The sort of price p[1...L], and M supplied service)

```
set J = {}; //assume that p[1...L] has been sorted by Algorithm 1
for each i ∈ [1, L] do
    if p_i > 0 && i ≥ M + 1 || p_i < 0 && i < M + 1 then
        J = J ∪ {p_i};
    end if
end for
return J;
```

Theorem 3. *The trading set under Algorithm 5 achieves a competitive equilibrium.*

Proof. i) Let T denote the trading set generated by Algorithm 5. Consider the trader who is at the (M+1)-th position is a seller. Let S denote the set of prices of sellers that are in T. According to the market clearing rule, we can obtain S by the following equation.

$$S = \{p_i \quad p_i > 0, | \quad rank(p_i) \geq M + 1\}, \tag{9}$$

where $rank(p_i)$ represents the position of p_i in T. Let $\overline{S} = \{b_i > 0 \quad | \quad rank(p_i) < M + 1\}$, then $|S| + |\overline{S}| = M$, where $|S|$ denotes the number of items in S and M is the total service under sell (including the sellers who are not in T). Let B denote the set of prices of buyers that are in T, then we can obtain:

$$B = \{p_i \quad p_i < 0, | \quad rank(p_i) < M + 1\}. \tag{10}$$

Since $\overline{S} \cup B = \{p_i | rand(p_i) < M + 1\}$, and $\overline{S} \cap B = \emptyset$, then $|B| + |\overline{S}| = M$. Also, because $|S| + |\overline{S}| = M$, therefore we can obtain $|S| = |B|$. That is, the number of prices of sellers equals the number of prices of buyers. As each seller (buyer) in T represents one supplied (demanded) unit, then the total supply equals to the total demand in T, which shows that the market achieves a competitive equilibrium.

ii) Consider the trader who is at the (M+1)-th position is a buyer, then according to the market clearing rule, the buyer price at the (M+1)-th position is not in T. Thus the buyer price set B is the same to that in (i). At the same time, we can compute the seller price set S by the following equation:

$$S = \{p_i > 0 \quad | \quad rank(p_i) > M + 1\}. \tag{11}$$

As the trader at the (M+1)-th position is a buyer, we can obtain:

$$S \equiv \{b_i > 0 \quad | \quad rank(b_i) \geq M + 1\}. \tag{12}$$

Similar to (i), we can show that the market achieves a competitive equilibrium as well. This completes the proof. □

3) Procedure 3 - Matching the traders: Given the trading set generated by Algorithm 5, we now describe how to match the buyers and sellers.

For the service trading model, there will be system-level cost for managing the trading. The type of the cost may vary depending on the specific technology to perform the trading, such as the computational and storage resources of the organizer. The trading system should minimize the trading cost so that the system is scalable enough to manage millions of users. Generally, we can expect that the more trading transactions there are the higher trading maintenance cost it will be. As a result, the system should efficiently match the traders so as to minimize the number of trading transactions. We now formulate this minimization problem.

Consider W is a trading set generated by Algorithm 5. Consider there are c sellers in W, $c \leq m$. Let $S_1, S_2, \cdots, S_c$ denote the quantity of service that are to be traded of the sellers. $\sum_{i=1}^{c} S_i = Q_S(P)$, where P is the market clearing price. Consider there are d buyers in W, $d \leq n$, $\sum_{j-1}^{d} K_j = Q_D(P)$. The matching problem can be formalized as the following linear programming problem:

$$\min \sum_{i=1}^{c} \sum_{j=1}^{d} \varphi(x_{ij}),$$

$$s.t. \quad \begin{cases} \sum_{j=1}^{d} x_{ij} = S_i, \quad i = 1, 2, \cdots, c, \\ \sum_{i=1}^{c} x_{ij} = K_j, \quad j = 1, 2, \cdots, d, \\ x_{ij} \in N, i = 1, 2, \cdots, c; j = 1, 2, \cdots, d, \end{cases}$$

where x_{ij} denotes the quantity of service unit between seller i and buyer j, and $\varphi(x_{ij})$ denotes the trading cost of selling (buying) x_{ij} unit of service.

Given the competitive equilibrium, we have $Q_S(P) = Q_D(P)$. In order to minimize the trading cost, we need to minimize the number of transactions. That is, we need to minimize the number of each non-zero x_{ij}, or maximize the value of non-zero x_{ij}. Because $0 \leq x_{ij} \leq min\{S_i, K_j\}$, the minimum number of transactions can be achieved if we set

each non-zero x_{ij} to be its maximum value ($min\{S_i, K_j\}$). The following procedure gives the solution to the above linear programming problem.

Matching Procedure. Given a trading set generated by Algorithm 5, the system first matches the seller who has the highest supply and the buyer who has the highest demand, the quantity of the trading is the minimum quantity between the seller's supply and the buyer's demand. This will lead to one of the following cases: 1) Both the seller and the buyer leave the trading set if the supply equals to the demand, otherwise 2) the trader with the larger quantity of supply or demand remains in the trading set. If (1) happens, the organizer simply repeats the matching process. If (2) happens, the organizer puts the remaining trader back to the trading set and repeats the matching process. As the trading set achieves a competitive equilibrium, the process will terminate in finite steps and the supply (demand) of all the traders can be satisfied.

The computational complexity of sorting sellers and buyers is $O(\max(M, M))$, while that of the matching process is $O(M + N)$. By combining the above procedures, we solve the auction problem with a computational complexity of $O(M + N)$, i.e., $O(L)$.

Note that, when M=1, the auction system is Vickrey's Second Price Sealed Auction. Vickrey proved it is strategy-proof [63]. When $M > 1$, we prove that such an auction model is strategy-proof for all buyers and risk averse sellers. By *risk averse* we mean that the trader is reluctant to accept a bargain with an uncertain payoff rather than another bargain with a certain and possibly lower payoff.

The following two theorems lay the foundation of TCDA's property, with proofs provided in detail in Appendix A and Appendix B.

Theorem 4. *TCDA is strategy-proof for buyers.*

Theorem 5. *TCDA is strategy-proof for risk averse sellers.*

In contrast to regular auction markets such as electricity and stock, an IoT device may dynamically join and leave the system. When such situation happens, Blockcloud can use a signaling message to let the remaining trader quickly join the next auction. The system can also use a fine-grained accounting approach that the buyer gradually pays tokens to the service provider. In this way, the buyer does not waste money if the seller leaves the network before the agreed quantity of service is consumed.

VIII. Conclusion

In this paper, we propose a new approach that combines the advantages of SCN and blockchain to empower IoT. We propose a service-centric blockchain architecture, which leverages a service-driven communication paradigm to support mobility and scalability. We design a new consensus mechanism and en efficient ledger structure to provide trust, security and incentive in a decentralized manner. Finally, we develop a service pricing and distribution mechanism that fairly protects the profits of both service providers and users of the network.

Appendix A
Proof of Theorem 4

Theorem 4: TCDA is strategy-proof for buyers.

Proof. Assume that the real estimated price of unit service of buyer j is v_j. Let b denote the auction price that buyer j submits to the auction. Suppose that the price of other traders will not change, so the result of the market clearing price y will only relate to buyer j at the end of the auction. Let β represent the market clearing price when buyer j submits the real price v_j (assume that v_j is the sth highest price among all submitted prices). Let β_{+1} and β_{-1} represent the (s+1)-th and the (s-1)-th highest price when buyer j declares price v_j. The utility of buyer is considered as the difference value between real price v_j and market clearing price y.

(i) When $\beta > v_j$, the relationship between the market clearing price y and the declared price b of buyer j is illustrated by Figure 12, where the shadow part represents the situation that buyer j can win the auction and the area marked with '-' means that buyer j gains negative utility when he or she wins the auction. The bold lines show the relationship of market clearing price y and buyer j's submitted price b when other traders' auction prices keep unchanged. We see that even through buyer j wins the auction by offering a price higher than v_j, the utility will be negative, which is $v_j - \beta_{+1}$. Therefor the best strategy of buyer j is to declare the real price even if buyer j can not win the auction. On the other hand, declaring a price that is lower than v_j will not help the buyer to win the auction or change the market clearing price, thus the buyer has no incentive to do so.

(ii) When $\beta < v_j$, the relationship between the market clearing price y and the declared price b of buyer j is illustrated by Figure 13, where the area marked with '+' denotes the positive utility when buyer j wins the auction. As shown in the figure, when buyer j declares the real price, he or she will win the auction and get strictly positive utility, which is $v_j\beta$. Although declaring a higher price could also gain the same utility in this case, doing this may increase the possibility of getting negative utility, which is shown in (i). If buyer j declares a lower price, the market clearing price might be reduced, but the buyer can not improve the utility as he or she will lose the auction.

(iii) When $\beta = v_j$, the relationship between the market clearing price y and the declared price b of buyer j is illustrated by Figure 14. In this case, the market clearing price is determined by buyer j's price. However, buyer j would not lower the price even if he or she knows this situation in advance. This is because, although declaring a lower price could lead to a positive utility, the possibility of winning the auction is zero (shown in Figure 6). This is not what a rational buyer would do. Also, buyer j would not choose to increase the declared price, because that will lead to a negative utility even if he or she can win the auction.

Based on the above analysis, no matter what the price of other traders are, in all cases, buyer j gains the best utility when he or she declares the real price. At the same time, only

by declaring the real price can the buyer maximize the utility in all the above situations. Because TCDA is based on sealed biding, buyers can not know which situation will happen, so the best strategy is to declare the real price. This completes the proof. □

APPENDIX B
PROOF OF THEOREM 5

Theorem 5: TCDA is strategy-proof for risk averse sellers.

Proof. Assume that seller i's marginal (real) price is u_i, and the price that seller i submits to the auction is b. Let γ represent the market clearing price when seller i declares the real price u_i, and γ_{+1} and γ_{-1} represent the (i+1)-th and the (i-1-th highest prices under u_i. Under the situation that other traders do not change their price, the relationship between the market clearing price y and seller i's auction price b is illustrated by Figure 15-17.

(i) When $\gamma > u_i$, the relationship is shown in Figure 15. We can see from the figure that when sellers declare their marginal price, they will win the auction and gain strictly positive utility $\gamma - u_i$. Although they can gain the same utility by offering a lower price, it can lead to the possibility of gaining a negative utility. Let the overlapping part of the shadow (including the border) represent the pricing area that the seller can gain a higher utility than submitting the real price, namely $\gamma \leq b \leq \gamma_{+1}$. When the seller increases the price (we only discuss the situation in which the market clearing price is changed), a higher utility might be achieved, but because the seller can not accurately forecast the value of γ and γ_{+1}, he or she may also get a lower utility. Specifically, if the seller concludes a wrong expectation and makes $b > \gamma_{+1}$, he or she will not only miss the expected higher utility, but also lose the stable income that comes from offering the real price. This is not the optimal strategy for a risk averse seller.

(ii) When $\gamma < u_i$, the relationship is shown in Figure 16. By declaring the marginal price the seller will not win the auction, so the utility is zero. The seller can choose to reduce the price to win the auction. However, even though the seller wins the auction, he or she will gain a negative utility. Therefor it makes no sense for the seller to reduce the price. There is no incentive for the seller to increase the price as it does not help the seller to win the auction.

(iii) When $\gamma = u_i$, the relationship is shown in Figure 17. In this situation, the seller could gain more utility by declaring a higher price than u_i. This is because the utility of the seller is zero under the real price. By declaring a higher price, the seller may either get zero utility by losing the auction ($b \geq \gamma_{+1}$), or get a strictly positive utility by winning the auction ($u_I < b < \gamma_{+1}$).

Although under situation (i) and (iii), there exists the possibility of getting a higher utility by declaring a higher price, the seller can not make it always happen by accurately forecasting γ and γ_{+1}. Under a wrong estimation, there can be the situation of $\gamma > u_i$, in which the seller will not only miss the expected excess utility, but also lose the stable income that is brought by offering the real auction price. That is not the optimal strategy for a risk averse seller, because the possibility of losing the stable utility $P(\gamma > u_i)$ is much higher than that of gaining a higher utility by declaring a higher price $P(\gamma \geq u_i)P(\gamma \leq b \leq \gamma_{+1})$.

Based on the above analysis, no matter what auction price other traders declare, in order not to increasing the possibility of winning the auction, the best strategy of a risk averse seller is to declare the real price.

This completes the proof.

□

REFERENCES

[1] (2017) Ericsson mobility report: On the pulse of the networked society. [Online]. Available: https://www.ericsson.com/mobility-report

[2] "Device democracy: Saving the future of the internet of things," in *IoT Report of IBM Institute for Business Value*, 2015.

[3] X. Sun and N. Ansair, "Edgeiot: Mobile edge computing for the internet of things," in *IEEE Communications Magazine*, vol. 54, no. 12, 2016, pp. 22–29.

[4] ——, "Dynamic resource caching in the iot application layer for smart cities," in *IEEE Internet of Things Journal,*, 2017.

[5] A. Sathiaseelan, L. Wang, A. Aucinas, G. Tyson, and J. Crowcroft, "Scandex: Service centric networking for challenged decentralised networks," in *Proceedings of the 2015 Workshop on Do-it-yourself Networking: An Interdisciplinary Approach*. ACM, 2015.

[6] M. Freedman, J. Rexford, S. Ko, P. Gopalan, and D. Shue, "A scaffold for wide-area distributed services," in *6th GENI Engineering Conference*, 2009.

[7] T. Koponen, M. Chawla, B. G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *Proc. of the ACM SIGCOMM*, 2007.

[8] Named data networking (ndn) project. PARC Technical Report NDN-0001. [Online]. Available: http://www.named-data.net/ndn-proj.pdf

[9] D. Griffin, M. Rio, P. Simoens, and M. Franke, "Service-centric networking," in *Handbook of Research on Redesigning the Future of Internet Architectures*, 2015.

[10] M. Gritter and D. R. Cheriton. (2000, Jul.) Triad: A new next-generation internet architecture. [Online]. Available: http://www-dsg.stanford.edu/triad/

[11] H. Balakrishman, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish, "A layered naming architecture for the internet," in *ACM SIGCOMM*, 2004.

[12] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayannan, I. Stoica, and S. Shenker, "Rofl: Routing on flat labels," in *ACM SIGCOMM*, 2006.

[13] A. Detti, M. Pomposini, N. Blefari-Melazzi, and S. Salsano, "Supporting the web with an information centric network that routes by name," *Computer Networks*, vol. 56, no. 17, pp. 3705–3722, 2012.

[14] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of information (netinf) c an information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, pp. 721 – 735, 2013.

[15] E. Nordstrom, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Ko, J. Rexford, and M. Freedman, "Serval: An end-host stack for service-centric networking," in *9th USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, 2012.

[16] A. Ghodsi, T. Koponen, B. Raghavan, S. Shenker, S. A, and W. J, "Information-centric networking: Seeing the forest for the trees," in *Hotnets'11*, 2011.

[17] L. Federico, A. Leonardo, D. Stefano, M. Andrea, and S. Vladimiro, "A blockchain-based infrastructure for reliable and cost-effective iot-aided smart grids," in *Living in the Internet of Things Conference: Cybersecurity of the IoT*, 2018.

[18] P. Sharma, C. M, and J. Park, "A software defined fog node based distributed blockchain cloud architecture for iot," in *IEEE Access*, 2018.

[19] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an optimized blockchain for iot," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, ser. IoTDI '17, 2017, pp. 173–178.

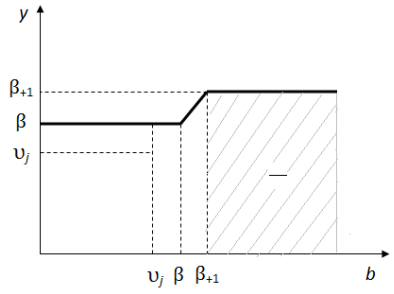

Fig. 12: The relationship between y and b when $\beta > v_j$.

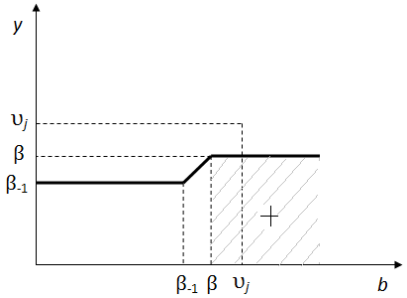

Fig. 13: The relationship between y and b when $\beta < v_j$.

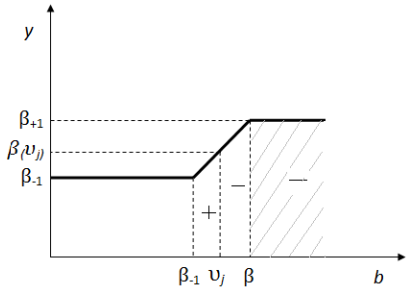

Fig. 14: The relationship between y and b when $\beta = v_j$.

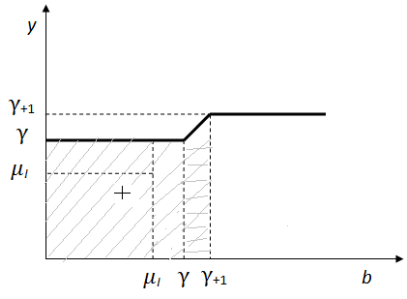

Fig. 15: $\gamma > u_j$

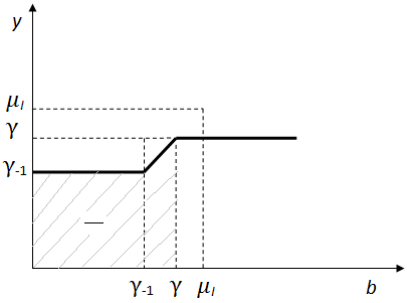

Fig. 16: $\gamma < u_j$

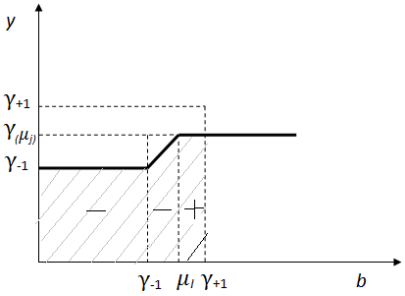

Fig. 17: $\gamma = u_j$

[20] O. Novo, "Blockchain meets iot: An architecture for scalable access management in iot," in *IEEE Internet of Things Journal*, 2018, pp. 1184–1195.

[21] M. Iansiti and K. Lakhani, "The truth about blockchain," in *Harvard Business Review*, 2017.

[22] D. Wörner and T. von Bomhard, "When your sensor earns money: Exchanging data for cash with bitcoin," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014, pp. 295–298.

[23] E. Alsaadi and A. Tubaishat, "Internet of things: Features, challenges, and vulnerabilities," in *International Journal of Advanced Computer Science and Information Technology (IJACSIT)*, vol. 4, no. 1, 2015, pp. 1–13.

[24] "Challenges in the internet of things iot," in *Texas Instruments (IT) IoT Report*, 2016.

[25] "Irtf information-centric networking research group (icnrg)," in *irtf.org*, 2016.

[26] J. Barr, "Amazon s3 two trillion objects, 1.1 million requests / second," 2013. [Online]. Available: https://aws.amazon.com/blogs/aws/amazon-s3-two-trillion-objects-11-million-requests-second/

[27] M. Freedman, E. Freudenthal, and D. Mazires, "Democratizing content publication with coral," in *1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[28] S. Yang, Z. Ming, D. Pan, K. Xu, and D. Wang, "An efficient and secure blockchain protocol for internet of things," in *Blockcloud Consensus Technical Report*, 2018.

[29] S. Popov, "The tangle version 1.3," in *IOTA White Paper*, 2017.

[30] Z. Ming, M. Xu, C. Xia, D. Li, and D. Wang, "Siona: A service and information oriented network architecture," in *Journal of Network and Computer Applications*, 2014, pp. 80–91.

[31] Z. Ming, J. Ubillos, and M. Xu, "Name-based shim6: a name-based approach to host mobility," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 15, no. 4, 2012.

[32] D. Freedman, "The double auction institution: A survey," in *Santa Fe Institute Studies in the Sciences of Complexity*, 1993, pp. 3–25.

[33] P. Vytelingum, D. Cliff, and J. N. R, "Strategic bidding in continuous double auctions," in *Elsevier Artificial Intelligence*, vol. 172, 2008, pp. 1700–1729.

[34] Z. Ming, M. Xu, N. Wang, B. Gao, and Q. Li, "Towards a truthful auction for user data allowance trading in mobile networks," in *37th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017.

[35] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang, "Smart data pricing: Using economics to manage network congestion," *Commun. ACM*, vol. 58, no. 12, pp. 86–93, Nov. 2015.

[36] "Filecoin: A cryptocurrency operated file storage network," https://filecoin.io/, 2017.

[37] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networking," in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, Aug 2014, pp. 1–8.

[38] S. Yang, M. Xu, D. Wang, and J. Wu, "Source address filtering for large scale network: A cooperative software mechanism design," in *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, July 2012, pp. 1–7.

[39] Z. Ming, M. Xu, N. Wang, B. Gao, and Q. Li, "Truthful auctions for user data allowance trading in mobile networks," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 1271–1280.

[40] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena, "Demystifying incentives in the consensus computer," in *Proc. ACM CCS' 15*, New York, NY, USA, 2015.

[41] J. Teutsch and C. Reitwiebner, "A scalable verification solution for blockchains," https://truebit.io/, 2017.

[42] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *OSDI, 1999*, 1999.

[43] "The bitcoin lightning network:," https://lightning.network/lightning-network-paper.pdf.

[44] 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.

[45] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[46] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," 2015.

[47] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-ng: A scalable blockchain protocol," in *USENIX NSDI' 16*, Santa Clara, CA, 2016.

[48] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," 2014.

[49] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Advances in Cryptology – EUROCRYPT 2017*, 2017, pp. 643–673.

[50] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *Financial Cryptography and Data Security*, 2015, pp. 528–547.

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

[52] https://byteball.org/Byteball.pdf, "Byteball: A decentralized system for storage and transfer of value," http://www.gartner.com/newsroom/id/3165317.

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

[54] 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.

[55] R. Pass and E. Shi, "Hybrid consensus: Efficient consensus in the permissionless model," in *DISC' 17*.

[56] S. Nakamoto, "the proof-of-work chain is a solution to the "byzantine generals" problem," https://www.mailarchive.com/cryptography@metzdowd.com/msg09997.html, Nov 2008.

[57] R. Pass and E. Shi, "Thunderella: Blockchains with optimistic instant confirmation," https://www.thundertoken.com, 2018.

[58] "Eos.io technical white paper v2," 2018.

[59] 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.

[60] D. Li, J. Wu, J. Liu, Y. Cui, and K. Xu, "Defending against distance cheating in link-weighted application-layer multicast," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 5, pp. 1448–1457, 2011.

[61] J. Huang, R. A. Berry, and M. L. Honig, "Auction-based spectrum sharing," *Mob. Netw. Appl.*, vol. 11, no. 3, pp. 405–418, 2006.

[62] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker, "A bgp-based mechanism for lowest-cost routing," *Distributed Computing*, vol. 18, no. 1, pp. 61–72, 2005.

[63] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.

[64] E. Clarke, "Multipart pricing of public goods," *Public Choice*, vol. 11, no. 1, pp. 17–33, 1971.

[65] T. Groves, "Incentives in teams," *Econometrica*, vol. 41, no. 4, pp. 617–631, 1973.

[66] P. Vytelingum, D. Cliff, and N. R. Jennings, "Strategic bidding in continuous double auctions," *Elsevier Artificial Intelligence*, vol. 172, pp. 1700–1729, 2008.

[67] P. D. Klemperer, "How (not) to run auctions: The european 3g telecom auctions," in *Working Paper, Oxford University*, 2001.