IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 11, 2022, accepted May 9, 2022, date of publication May 18, 2022, date of current version June 10, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3176349

Function Virtualization Can Play a Great Role in Blockchain Consensus

JUN WANG[1], JIANG ZHU[2], MINGHUI ZHANG[3], IQBAL ALAM[4], AND SUJIT BISWAS[5,6], (Member, IEEE)

[1]30th Research Institute of China Electronic Technology Group Corporation, Beijing 100846, China
[2]Graduate School, José Rizal University, Mandaluyong City, Philippines
[3]School of Information Engineering, Guangdong Innovative Technical College, Dongguan, Guangdong 523960, China
[4]Nan Yang Academy of Sciences (NASS), Beijing 102400, China
[5]Electrical and Electronics Engineering Department, University of Surrey, Guildford GU2 7XH, U.K.
[6]Computer Science and Engineering Department, Faridpur Engineering College, University of Dhaka, Dhaka 1000, Bangladesh

Corresponding author: Jiang Zhu (jiangzhujohn@qq.com)

ABSTRACT Bitcoin introduced a cryptocurrency as a form of public ledger consequently that turned into a most popular security technology, Blockchain. Its integrated mining technology lies the key security mechanism. The system allows forming a pool mining group to solve a particular job and share their revenues to their CPU usage while one of them successfully mines a block. To mine a block, a cryptographic puzzle should be solved, which requires significant compute resources that cause huge energy consumption. On the other hand, recent statistics show that low computational energy-restricted Internet of Things (IoT) devices are increasing exponentially. Although it has low energy and limited computation power, it is large in quantity when it is integrated. So we focus on a stochastic geometry theory, which resolves the issue of block mining computation via utilizing multiple mobile IoT devices, given that these IoT devices are Computation Capable Nodes (CCNs). To further normalize this issue, we propose an efficient mathematical solution that uses smart coordination of Virtual Network Functions (VNFs) for IoT devices to enable their CPU usage efficiently. At the same time, the work and credit point distribution policy is smartly handled through virtual pool mining. The proposal renders Network Function Virtualization technology to configure VNF, and Service Function Chain technology is utilized to enable the network flow of such VNFs. New algorithms are presented to solve multiple issues like node discovery, computation offloading, and work credit point distribution. Our goal is to minimize energy consumption within the given time constraint. Implementation results show that although virtual functions for block mining require extensive computations in IoT devices, dividing computation work into small fractions called tasks embedded with VNF, and offloading them to nearby CCNs, tend to minimize the cost and energy consumption of individual shared miners. The overall mining process is proved efficient and faster.

INDEX TERMS Internet of Things, network function virtualization, virtual network function, service function chain, blockchain, bitcoin, pool mining.

I. INTRODUCTION

The term consensus refers to a mechanism where ~~by~~ a group of nodes in a Peer-to-Peer (P2P) network dynamically reach ~~to~~ a pre-defined agreement among themselves to solve a cryptographic solution. The first node to find this solution successfully forms a block using a valid algorithm called Proof of Work. Series of such blocks hold valid transaction records in a ledger distributed across all peers in a P2P network called Blockchain (BC). There are different kinds of consensus mechanisms such as Proof-of-Work (PoW), Proof-of-Stake (PoS), Delegated Proof-of-Stake (DPoS), Proof-of-Authority (PoA), Proof-of-Weight (PoWeight), Byzantine Fault Tolerance (BFT), etc. Unique fault-tolerant consensus algorithms solve many challenging issues, such as the double-spending problem in a decentralized network. Consensus participants follow different strategies to eventually come to a single opinion to approve a block. The strategy is well-known and introduced as mining through Bitcoin [1], and the participants are recognized as miners. More than two thousand cryptocurrencies [2] are present in the market after the revolution of the Bitcoin economy. Due to the public

The associate editor coordinating the review of this manuscript and approving it for publication was Daniel Grosu.

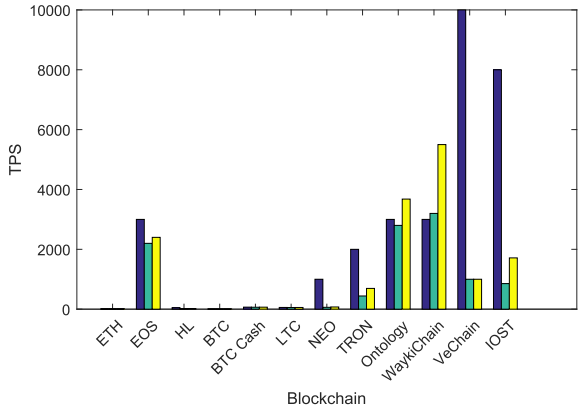


FIGURE 1. Comparison of TPS in different blockchain.

nature and scalability of public blockchain (i.e., cryptocurrency), recent technology has gone beyond cryptocurrency, also called distributed private BC. It has various application domains, such as IoT, E-healthcare, education, banking, etc.

Although the nature of BC varies in different use cases, there remain some standard features like a consensus, distributed ledger, etc. Every BC is primarily different based upon their nature and core technical features of consensus algorithms whose ultimate goal is to improve TPS without compromising security. Figure 1 refers to comparisons of different types of BC concerning TPS. The blue, green, and yellow bar denotes the official release of TPS, tested average TPS, and tested peak TPS, respectively. This indicates a contradiction in the Quality of Service (QoS) assured for users.

In a BC network, a block is added to the ledger in every certain time interval (e.g., 10min/block in bitcoin) against valid PoW [3]. A valid PoW is an integral part of a cryptocurrency network that solves some cryptography problem (i.e., mathematical puzzle). As a result, it allows a block of pending transactions to be added to the BC (i.e., records of all transactions ever). The integral part must be trivial to verify whether data satisfies the said requirements, e.g., difficulty value must start with a certain number of zeros to make the puzzle complex and resource-intensive for miners called hashing. Every round, a new hash is generated from the incremented nonce. This process assures that the number of blocks found each day remains steady, and more computing power is needed to mine the next block. It indicates that it is practically impossible to mine anything significant without having specialized hardware. At the same time, we are surrounded by billions of mobile IoT devices that could replace the utility derived from specialized hardware. Users of Mobile IoT devices like laptops, android phones, iPhones, iPads, etc., would reach 75.44 billion users worldwide, and each person is expected to be connected to 6 such IoT devices by 2025 [4]. Owners of vast businesses tend to be users of IoT devices to make their business automated and highly dependent on it. So more business transactions are becoming associated with IoT devices. A new sensing era lies in Mobile crowdsensing (MCS), which explores the advantage of extensive use of mobile IoT devices to collect data efficiently and enable several significant applications. It uses sensing and wireless communication potentials ~~to provide~~ services to billions of mobile IoT devices. The use of such IoT devices can be extended for BC applications too. BC mining requires computation-intensive tasks for miners, i.e., solving the PoW puzzle, which demands high computational capability and storage facilities from miners' perspective. Instead of buying new expensive server machines, it is possible to minimize computation costs and optimize resource usage by utilizing these idle IoT devices for BC mining purposes. Since IoT devices tend to be low-power resource-constrained devices, a single IoT device cannot replace the utility of a high-end server machine. One efficient technique to mitigate this issue is to divide the computation work into a fraction of small tasks and offload them to multiple available mobile IoT devices. Only specific IoT devices are preferred to be utilized for mining purposes based on their computational capability, and we call them CCNs. But finding optimal mining policy while Computation Offloading (CO) in wireless mobile BC networks is more challenging compared to BC mining in the wired network because of extensive computation mining process [5]. On the other end, function Virtualization (FV) addresses this issue by dividing the computation extensive mining process of a single miner in wireless mobile BC networks by sharing partial mining work via FV with nearby mobile users where FV exploits the virtualization paradigm.

Virtualization [6] is the logical abstraction of the underlying hardware devices within a network through software implementation. The abstraction decouples the control from hardware and makes it easier to modify, manage, and upgrade. In recent times, the abstraction has not been limited to hardware only. Still, rather software embedded into hardware has also been virtualized as independent elements, which is referred to as Function Virtualization [7], [8]. To this end, FV is implemented through VNF, which renders Network Function Virtualization (NFV) [9] technology. NFV flexibly utilizes network and computation resources [10]–[12]. It reduces mining expenses (e.g., capital and operational expenditures, CAPEX, OPEX [13]), computation delay. It also minimizes energy usage for any mobile operators via configuring software instances (i.e., VNFs) into assigned server/edge nodes rather than configuring Network Functions (NFs) on dedicated hardware [14]. Series of VNFs are interlinked via virtual links (VLs), forming a chain called Service Function Chain (SFC) [8]. NFV rendering SFC alleviates the overall traffic burden both in core and edge network because mobile users' CO tasks can be randomly configured into series of VNFs [9] and distributed to multiple CCNs. Each executes one fraction of the entire work in parallel, eventually saving time. CCNs have higher computation capability than resource-constrained IoT devices like headphones, smart shoes, smartwatches, etc. They can execute multiple computational tasks in parallel, eventually serving

the purpose of FV for miners. Nowadays, people regularly use too many CCNs, which are multitasking capable. It is highly possible to share mining tasks through FV to those capable devices and interested in participating in the mining process. This will allow a huge number of participants to strengthen the effectiveness of the overall consensus process. This process can also cut energy consumption due to reduced computation usage in every mobile user. The entire work is now jointly solved via multiple mobile devices rather than executing it in one single node. Hence, the entire process will become faster and more efficient than the traditional process/block mining strategy because of the dedicated virtual function for each offloading task. There are no BC-based papers focusing on the virtualization technique for CO to nearby devices. So we are only able to study papers focusing on mining policy management for BC networks and CO schemes for Mobile Edge Computing (MEC). Houy [15] presents a non-cooperative game among multiple miners, where each miner can choose to include the number of transactions in a block. Fisch and Shelat [16] focuses on the propagation of mined block using both sequential and stochastic game theory. Reference [17] optimizes the pool mining mechanism via cooperative game-based BC mining. Luu *et al.* [18] presents a unique game for computational power split while BC mining, where miners can earn rewards by solving the PoW puzzle. Optimization of offloading ratio, speed of computation, and rate of transmission are jointly scrutinized in [19]. Mao *et al.* [20] focuses on optimizing offloading schemes, Yu *et al.* [21] focuses on optimal CPU time allocation, and Bi and Zhang [22] optimizes the trade-off between local computing and offloading. While Kang *et al.* [23] exploiting virtualization technique investigates a trade-off between device-to-device (D2D) reliability and computation load for every server through collaborative design of SFC and forwarding graph embedding. Kwak *et al.* [24] focuses on minimizing CPU and network energy of mobile devices considering a given delay threshold while mobile cloud offloading. In contrast to FV for CO while block mining, we strongly believe there is still scope to contribute to this field. The contributions of this paper summarizes in the following points.

- We have proposed a novel block mining framework for public blockchain network where lightweight IoT devices play a miner role. IoT devices utilize NFV and SFC technology to address extensive computation for PoW mining mechanisms.
- It introduces joint mining computation modes to avoid the overload of a single miner.
- CO schemes rendering NFV framework are jointly derived as an optimization problem. Our goal is to minimize latency, cost, bandwidth consumption while optimizing resource allocation within a given time constraint.
- We propose virtual mining, which allows task sharing among registered capable miners via executing corresponding VNFs for each task. This part utilizes NFV technology to configure VNF for task execution and further link them using SFC technology to accumulate the execution result of various VNFs, eventually finding the full PoW of a given work.
- Stochastic geometry theory is used theoretically to derive the related performance metrics that include energy consumption and delay.
- We further explain the complexity of computation offloading using a dependency graph.
- Finally, we show that FV can reduce computation cost, minimize energy consumption, maximize net revenue, and make faster mining.

II. LITERATURE REVIEW AND MOTIVATION

Because of security concerns, BC is becoming more popular day by day, and users are increasing numerously. Many industries have already started to do their business transactions via BC like systems, and many more are thinking of adopting a digital currency system for doing business transactions. The high cost of mining and scalability are significant constraints of the rapid expansion of the technology. Due to computational limitations, not all IoT devices can be considered directly as BC nodes [25]. Typical miner should have a large computation capacity and require a massive amount of energy for PoW. Both technical features are constraints for IoT being a minor. IoT devices are increasing in an unprecedented way. Although they have limited computation power, successful resource sharing of these large quantity devices can put the power in balance. This issue arises by targeting the optimum solution. Primary research already shows that orchestration of VNF can minimize power consumption [26] for CO. At the same time, the mining work is distributed within short coverage to multiple mobile devices (i.e., CCNs). This means less computation usage in a single node, as it only requires executing a fraction of the entire work. Since multiple fractions of the whole mining work can concurrently run in numerous CCNs, it would need less time to complete the entire work than a single node executing the same work alone. Hence computation is more efficiently managed by the NFV framework. Eventually, it will scale among all business BC nodes and help the BC transactions to execute efficiently without forgoing security. As more BC nodes can participate in the voting process, it can strengthen the weight of the consensus formation. Below are some of the existing solutions that pertain to BC mining and FV strategies for CO, separately.

A. BLOCKCHAIN MINING STRATEGIES

The articles [27]–[31] elaborate different aspects of BC mining where Kano and Nakajima investigates the mining work concentration issue in [27], and proposes an incentive-based solution considers psychological factors via gamification instead of offering economic incentives. The solution also uses virtual currency service for users, which is only limited to gaming purposes. Lin *et al.* [28] proposes a sustainable rewarding mechanism for block mining in a BC-based

transaction system. A secure transaction fee collection algorithm is proposed to enhance the reward distribution mechanism after successful mining, which can be considered as the key benefit of the solution. Simulation output shows that the proposed protocol implementation can steady block mining and reward distribution, but does not consider multi-pool scenario while block mining (i.e., limitation). Kiayias *et al.* [29] studies and formulate the stochastic game theory to decide a miner which blocks to mine and when to release blocks so that other miners cannot continue mining from it. One limitation is that it requires considerable computational power for mining rigs. Bae and Lim [30] validates the security aspect of a miner while mining rigs by proposing a mathematical random mining group selection mechanism to reduce the probability of successful double-spending attacks. Abe *et al.* [31] describes that mining power is relavent on the network and price of tokens that can be taken securely on a BC. Users exchange tokens on the PoW, while BC should monitor mining power and allow exchange tokens cheaper than the attack cost so that profit and cost of the attacker are not in equilibrium. Many recent contributions suggested consensus mechanism avoiding the complexity of mining [32], [33]. However, these contributions focused on private Blockchain and can not be applied to public Blockchain, making this contribution different.

B. FV STRATEGIES FOR OFFLOADING

Virtualization techniques regarding CO has been discussed in [9], [34], [36] where they focus on decoupling the control from the hardware to edge server by offloading network functions into software. Taking advantage of virtual Mobile Edge Computing (MEC), Chen *et al.* [34] considers the slicing mechanism of Radio Access Network (RAN). A network consisting of multiple base stations (BSs), where certain available BSs are selected to optimize computing tasks using Markov Chain [38] theory for offloading process between a mobile user (MU) and BSs. The solution renders a deep Q-network (DQN) strategy to learn optimal policies in dynamic networks. The Q-function [39] of the utility function is further combined with double DQN to propose a novel learning algorithm for resolving stochastic CO. Although numerical results show that the proposed solution performs better in delay and queue optimization, the exploitation of virtualization is limited to virtual network slicing only. Similarly, Sun *et al.* [40] places virtual computing resources to optimize storage resources for MEC in the mobile edge network (MEN). The solution uses Genetic Algorithm [41] for offloading decisions and an Openstack controller for configuring virtual instances. It also focuses on minimizing service delay while offloading tasks. Although simulation results show significant efficiency in utilizing storage resources by deploying virtual machines (VMs) in edge servers, the effort is limited to cost and delay optimization. However, Cheng *et al.* [35] uses virtualization in wireless networks that abstract multiple MEC nodes are providing efficient application service to both mobile users and MEC nodes. The solution exploits virtualization techniques to minimize delay and energy consumption cost and uses a heuristic algorithm [42] for CO decisions. The simulation results show that the heuristic approach outperforms fixed offloading while maintaining effective latency control. Cheng *et al.* Further using similar MEC technology, In [36], [37] proposed a solution on green computing in a small cell, and another on efficient power allocation that renders virtualization technology and distributed CO strategy, respectively. The solution adopts probabilistic Service Function Chain (SFC) and NFV for providing virtualization service and interlinks between them via virtual links and Management and Network Orchestration (MANO). To minimize the cost of CO in heterogeneous RAN, the solution also adopts integer linear programming to reduce the latency constraint, enhance resource allocation, and allow flexible use of applications for mobile users. The use case differs from our proposal because we consider CO for BC mining while they consider CO for gaming purposes only. They do not consider the BC network Performance evaluation, shows cost minimization between memory and computation usage and mobile users using the application. This solution provides optimal output with low complexity and a suitable environment for a large-scale network, which can be considered key benefits. Deployment of the proposed framework in heterogeneous RAN is challenging, and policy implementation to manage interference in such network considering end-to-end small BS communication is not considered and can be regarded as future work.

In light of the literature review in this section, we believe there are still gaps in optimizing the use of physical storage resources reducing latency and cost for resource-constrained mobile IoT devices. So BC mining becomes challenging for such IoT devices. Motivated from these issues, we propose smart coordination of VNF for CO, while computation work and reward distribution policy are smartly handled through virtual pool mining. The effort is included in Section III.

III. SYSTEM ARCHITECTURE

NFV based system model for block mining is illustrated in Figure 2. The variables used to formulate mathematical equations are given in Table 1.

This section have been categorized into six subsections where *system model* presents how a lot of CCN devices can be a part of BC mining network; *device-2-device communication (D2D)* describes how devices interact with each other in a heterogeneous environment, *system components* illustrates the features of various components of system model, *virtual functionalities* describes how MANO orchestrates VNF, *computation offloading* describes the offloading decisions, *preliminary metrics* formulates the elementary metrics of

TABLE 1. Variables used to formulate equations.

Abbr.	Des.	Abbr.	Des.
$H(T)$	Full work	T, t_i	Total number of tasks and individual task,respectively
T'	Tasks locally processed in m^l	$\bar{T}$	Tasks remotely processed in m_x
v_n	Corresponding VNF against T_n parts	G_n	Computation energy usage to process t_i in CPU circles/sec
D_n	Data size required to process t_i in bits	τ_n	Completion deadline time to process t_i in second
m_x	x participants who process $H(T')$ tasks with corresponding VNFs	$\boldsymbol{X}_n$	the size of the cryptographic hashes of blocks w.r.t. T parts
m_i	Solo miner who process $H(T)$ work locally in a BC network	m_j	Individual participant in m_x
m^l	Mining leader who locally process $H(T)$ tasks & offloads $H(T')$ waiting to find $p^{(PoW)}$	f^{m_x}, f^{m^l}	sum computational capability of miner m_j, m^l in a pool, respectively
κ^{m_x}	Computational energy coefficient of m_x	κ^{m^l}	Computational energy coefficient of m^l
α	The pathloss exponent	μ	Mean value of rayleigh fading
ϱ^2	The noise power	B	The bandwidth
P_u & P_{AP}	Transmit power of a miner & AP	P_S	The static circuit power of a miner
λ_e	Unit price of energy per Joules(J)	Q_n	number of CPU cycles to be executed in the task buffer of a miner
Υ^{m_x}	The reward for the miners to process $H(T')$ tasks in $mode$ 1	Υ^{m^l}	The reward for the miners to process $H(\bar{T})$ tasks locally in $mode$ 0
σ	Computation offloading decision vector	L_{VNF}	The capacity of D2D links between m^l, m_x
ϕ_u	Homogeneous Poisson Point Processes of m_i	ϑ_u	Density of users
p^{PoW}	Partial Proof of Work found in m_j	f^{PoW}	Full Proof of Work found in m^l
τ^D	Processing delay in $mode$ 0	τ^{D_1}	Processing delay in $mode$ 1
E^{m^l}	Energy consumption of m^l	E^{m_x}	Energy consumption of m_x
τ_N	time threshold/latency to discover neighbor node	Δ	Communication range between two nodes
(x^l, y^l)	position coordinates of m^l	(x_i, y_i)	position coordinates of m_i
$s^{\boldsymbol{t}}_{m^l}$	Sequence state of m^l in time $\boldsymbol{t}$ slots	$p_{\boldsymbol{t}}$	Transmission/Listening state probability
Δ	Duty cycle in time $\boldsymbol{t}$ slot that a node keeps its radio signal on	$\bar{n}$	Expected number of nearby CCNs
CO	Computation Offloading	CCN	Computation Capable Node

Abbr.=Abbreviation, Des.=Description

CO, and *problem formulation with constraints* elaborates the optimization objectives of CO scheduling.

A. SYSTEM MODEL

The system model supports BC technology based upon AP network with multiple mobile nodes within its network coverage. A peer node, in our context, is a device capable of

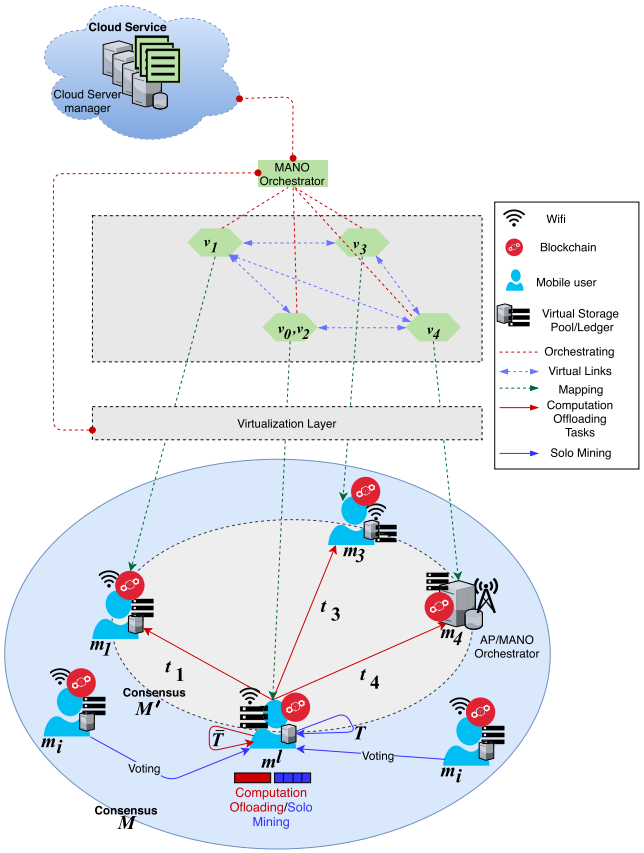


FIGURE 2. Virtual BC mining System model for extensive CO.

performing computation required for mining, also referred to as CCN. Every node in the BC network carries a unique identity *id* and can perform an individual task. A node performs a task for a specific period, and the node that performs such a task is a miner. Successful task execution is followed by both Partial Proof-of-Work (p^{PoW}) and Full Proof-of-Work (f^{PoW}). f^{PoW} occurs when a single miner successfully solves the mathematical puzzle of the entire work. At the same time, p^{PoW} occurs when a single miner divides complete work into a series of tasks to be solved by other individual registered miners concurrently. Every task contains a mathematical puzzle that a selected CCN must solve. Collective aggregation of p^{PoW} from such miners comprises of f^{PoW} signifies that a block is added to the BC network successfully, and reward is obtained. We presume that all CCNs can execute certain tasks assigned to them, as their computational capability is already weighted. At the same time, they agree to join the BC network. We presume that the list of available miners is obtained through the neighbor discovery process elaborated in section III-E, and such that the mining power of a single miner can be split and transmitted to multiple CCNs without computational constraints. Lets take there are A access points (APs), and n miners participating in a pool following two independent homogeneous Poisson point processes $\phi_a = \{AP_1, \ldots, AP_A\}$ and $\phi_u = \{m_1, \ldots, m_n\}$ with density ϑ_a and

ϑ_u, respectively, shown in Figure 2. We denote that one participant is considered an edge AP having enough capabilities to orchestrate MANO. Multiple pools may exist within a small network, but we consider only one pool in this network.

So associated miners with the nearest edge AP AP_m is denoted as $\phi_u = \{m_n\}$, where $\{m_n\} \in \phi_u, M = \{m_1, \ldots, m_n\}$. A work, initially, is evoked by a mining leader m^l, consist of a set of T tasks, where $T = t_i | \{i = 0, 1, \ldots, n\}$, which it tries to compute by itself. Every associated miner is taking part in the shared mining process which needs to input data with data size D_n (regarded in bits), have completion deadline time τ_n (regarded in second), and need computation capability G_n (regarded in CPU cycles per second). The decision to or not to offload computation is based upon the computation capability of m^l miner. We run a unique algorithm to compare the computation capability of the miner and the computation required to complete the work. While the algorithm determines m^l's computation capability is not enough to complete the work alone, we presume it is able to compute certain amount of $\bar{T}$ tasks as $\bar{T} = \{t_0, t_2\}$, $\bar{T} \subset T$, and the rest is offloaded to associated miners participating in the computation process. t_0 must reside on m^l locally validating the f^{PoW} found in M, and t_2 is an extra task that it may also compute locally without computational constraints. We denote the computation capability of m^l and AP_m as f^{m^l} and f^{AP_m}. In case of CO, we presume m^l offloads T' tasks to M' such that $T' = \{t_1, t_3, \ldots, t_x\}$, $M' = \{m_1, m_3, \ldots, m_x\}$, and $T' \subset T, M' \subset M$. The offloading is efficiently managed by configuring VNF for every offloading tasks, such that $V = v_i | \{i = 0, 1, 3, \ldots, n, n+1\}$ represents a set of VNFs corresponding to T', except 0 and $n+1$ denotes the first and last VNF residing on m^l validating the p^{PoW} found in M'. E.g. m^l offloads t_1 task to m_1 miner and assigns v_1 VNF that should be downloaded from AP_m in order to compute t_1 task locally by m_1 miner, and the same process applies for the rest of the miners taking part in the shared task computation process, shown in Figure 2. Else, we consider no function virtualization while the solo miner m^l solely hashes the $H(T)$ full work while associated miners (i.e m_i) only vote to form a consensus.

B. DEVICE-2-DEVICE COMMUNICATION

CO for block mining consists of various VNFs, and a user (i.e., m^l) of the BC network requests a set of VNFs to the orchestrator via application proxy, based on the amount of mining work. The network flow of VNFs is formed as SFC. SFC renders SDN and NFV capabilities to create a service chain of connected network services among firewalls, network address translation (NAT), intrusion protection, etc., eventually creating a virtual chain. SFC can configure many VNFs connected in an NFV environment, while its programmable interface allows customizing policy implementation via softwarization. Softwarization allows flexible software instances in virtual circuits that can easily set connections up or torn down as needed with the service chain provisioning through the NFV orchestration layer.

From the perspective of heterogeneous BC network for resource constrained IoT devices, distributed miners (i.e. M') are located within the coverage area of m^l. m_j is denoted as each CCN selected for CO, a set of m_j and an m^l together provide computation and storage facilities, i.e., $m_j \in M'$ are providing resources for virtual mining, and VNF components are deployed in the virtualization layer. Wireless virtual links are deployed between m^l and individual m_j of set M' for communication. m^l provisions application proxy to process service requested from M', a set of distributed miners. The application proxy is responsible for generating SFC by analyzing the information request of m^l for CO. The orchestrator, located in the edge network, is called Access Point (AP). It is responsible for deciding the optimal placement of VNFs forming SFC. The decision is updated to the proxy, and the orchestrator requests to initiate VNFs to each m_j. m_j provides computation resources for processing VNFs and communicates through wireless links during block mining. The wireless connections among associated miners of M', and between M' & m^l are mapped as virtual links (VLs) on the virtualization layer.

C. SYSTEM COMPONENTS

There are four different kinds of nodes in the system scenario serving other purposes in forming a BC network. They are leader miner (m^l), distributed miner (m_j), access point (AP_m), and cloud server.

1) LEADER MINER

All miners in the BC network may form a network of m_n, where individual m_i miners compete to hash T work successfully and try to find f^{PoW}. The first m_i to solve the f^{PoW} is referred to as m^l while others vote to form consensus, eventually m^l adds the new block in the ledger and earns the reward. It is called solo execution. It occurs only when m^l is capable of processing the full work based on its computational capability, and else it decides to offload task computation. Since it cannot hash T work by itself, it may decide to compute $\bar{T}$ tasks locally, as it is capable of, and distribute the rest of the T' tasks to nearby miners forming a virtual pool, where it remains the mining leader. The CO task distribution process renders SFC & NFV technologies efficient resource optimization service to mobile users (i.e., CCNs). With the help of the MANO component, which orchestrates VNFs, m^l transmits a series of offloading tasks to nearby CCN miners (i.e., M') in the pool along with state information of VNFs via VLs. Every offloading task has a corresponding VNF, which should be processed by individual m_j locally.

2) DISTRIBUTED MINER

Every m_j miner in M' is referred as a distributed miner. But it competes to solve a single off loadable task transmitted by m^l, and else solo execution occurs, which only requires voting from a set of m_i miners to form consensus, followed by successful hashing of T full work. Whether m_j miner may agree upon processing task computation, its work interest

depends on its computing capabilities. And hence successful hashing of each task represents f^{PoW} for m_j miner, but for the same corresponding task, m^l receives a p^{PoW}, considered as part of the full work processed.

3) ACCESS POINT (AP)

At the edge of the BC network, AP is considered when solo mining takes place, but it also participates in the mining process where needed. The only fact that differentiates AP from other miners is that it is the only node capable of orchestrating VNFs via MANO component, exploiting SFC with NFV principles. SFC provides services to store a series of VNFs and interconnects them. This allows an easy way of aggregating the result of multiple p^{PoW} found in M', eventually enabling an m^l to find f^{PoW}. Both m^l & m_j nodes render MANO's functionalities to obtain VNFs, while m_i doesn't.

4) CLOUD SERVER

A cloud server may provide backup storage services. Nodes in M are following common consensus as they compete to find f^{PoW}, while nodes in M' are following precise consensus with extended virtual functionalities, as they compete to find p^{PoW}. The replicated ledgers are stored in the cloud server once a block is successfully formed, as it is also part of the same BC network.

D. VIRTUAL FUNCTIONALITIES

CO and successful block mining is aided via SFC, as presented in Figure 3. The detailed process is divided into two parts: VNF placement section and mining related CO section.

In the VNF placement section, communication occurs between m^l & m_j, $m_j \in M'$. Let's presume G_{SFC} is the directed graph of SFC, which has a vertex set V. Loading all these VNFs consume energy & cost. We derive VNF load CPU usage in Section IV-B2. Initially, the proxy located in m^l receives service requests to generate an SFC for BC-based CO as per information provided by m^l. Given that m^l already has the list of available registered neighboring CCNs elaborated in Section III-E, where $\phi_u \cong p_\tau^{m^l}$. This process defines the strategy of node discovery of nearby CCNs. Then the proxy generates the requested G_{SFC} and forwards it to the orchestrator (i.e., AP_m) to generate VNFs. The respective CCN downloads the associated VNF as per its computation capability. VNF placement algorithm is proposed in Algorithm 3. The VNF placement is based upon the objectives of CO and its resource information of M'. With the produced result, the orchestrator forwards the decisions to m_j to initiate VNF downloaded from AP_m. Individual m_j executes one corresponding VNF, assigned for hashing one single task, and the orchestrator updates the decision placement to the proxy, shown in the VNF placement section of Figure 3. Then the proxy updates the decision response to m^l, the VNFs are initiated as per the service requested.

In the second section of Figure 3, it is noticed that m^l initially generates v_0 to provision the call sequence of the rest of the VNFs (i.e. $v_1, v_3 \ldots, v_n, v_{n+1}$), then starts forwarding

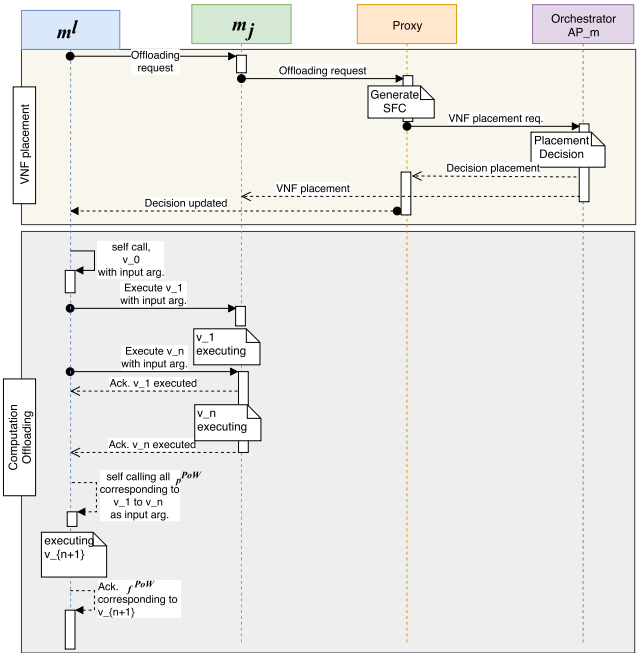


FIGURE 3. CO provision process for BC mining embedded with SFC.

it to AP_m. The call sequence V_{SFC} is presumed to have VNFs ranging from $v_0, \ldots, v_{n+1}$. m_1 downloads associated v_1 VNF with its input argument, executes v_1 locally, then returns the state information to m^l. This process is carried on until v_n is completed, where n is the $n^{(th)}$ term of the VNF to be processed in M'. Finally, m^l aggregates all the p^{PoW} via v_{n+1}, corresponding to the state information of each VNF from $v_1, v_3, \ldots, v_n$ which were executed in M'. While m^l eventually finds the f^{PoW}, shown in the above figure, corresponding to v_{n+1} state information, such that the objectives of the input/output arguments of v_0 matches with v_{n+1} (i.e. $v_0 \cong v_{n+1}$). Given that v_0 & v_{n+1} is the first and last term VNFs generated, to manage the placement of $v_1, v_3, \ldots, v_n$ residing on $m_1, m_3, \ldots, m_x$, respectively, and that the outcome of each VNF successfully hashes every individual task from t_1, t_3 to t_n with one task executing in every individual m_j miner. It is presumed that all m_js have equal computation capability, and can compute minimum one task assigned at a time. It is noticed that placement of VNFs are based on the application mining decisions, and the modeling of mining decisions are presented in Section IV-A.

E. NODE DISCOVERY AND REGISTRATION

This section introduces a Neighbour Discovery process of CCNs in a large-scale network. Communication may be well distributed when multiple nodes try to connect simultaneously, and transmission may fail due to interference. So we adopt a simple protocol model (derived from [21], [22]) to elaborate the process. We initially presume a node m^l can receive m_i's message signal successfully only if m_i is one of the transmitters within the m^l's single-hop communication range. As the CCNs are energy- and computation-constrained nodes, they can turn on or off their radio signals when needed

to save energy [17], [24]. Recalling from System Model (i.e., III-A), a set of nearby nodes are denoted as $M = \{m_1, \ldots, m_n\}$, where there exists a set consisting of n number of nearby nodes at the edge network distributed in a large area community via a fixed wireless channel. We presume a node's radio signal range has 10 *meters* $\leq \Delta \leq$ 100 *meters*. As the task is distributed sequentially, we consider the transmission of one task at a time. Lets presume there are two neighboring m^l, m_i nodes lying within a suitable distance $d(m^l, m_i) \leq \Delta$. Node m^l is well aware of its position (x^l, y^l) and computing local density using decision vectors:

$$f(x^l, y^l) = \begin{cases} \vartheta(x^l, y^l), \ where\ (x^l, y^l) \in D \\ 0, \ where\ (x^l, y^l) \notin D \end{cases} \quad (1)$$

where $\vartheta(x^l, y^l) \in \vartheta_u, D$ refers to network coverage area when m^l's distance from m_i is denoted as:

$$(x^l - x_i)^2 + (y^l - y_i)^2 \leq \Delta^2$$

given that Δ^2 must remain within the network coverage of m^l. m^l's expected number of nearby neighbors are derived as:

$$\bar{n}_{m^l} = m_x \pi \Delta^2 \vartheta(x^l, y^l)$$

Node m^l can discover its neighbor node m_i within τ_N time slot, and m_x is the number of CCNs available for task computation, as already mentioned in system model, and that m_i being one of the selected neighbor of m^l to transmit, given that m^l is listening in the slot. A pre-defined sequence $S^{m^l} = s^{m^l}(\tau), 0 \leq \tau \leq N$ of period N is scheduled for node m^l, in which:

$$s^{\tau}_{m^l} = \begin{cases} sleep\ m^l\ in\ time\ \tau\ slot\ radio\ signal\ turned\ off \\ transmit\ m^l\ in\ time\ \tau\ slot\ transmitting \\ listen\ m^l\ in\ time\ \tau\ slot\ listening \end{cases} \quad (2)$$

Duty cycle (θ) is the fraction of time that a node keep its radio signal turned on, formulated as:

$$\theta_{m^l} = \frac{|\tau : 0 \leq \tau \leq N, s^{\tau}_{m^l} \in transmit, listen|}{n} = 1 \quad (3)$$

for any m^l and m_i nodes, because we presume they have the same duty cycle at all times (i.e. $\theta_{m^l} = \theta_{m_i} = 1$). So they are referred as symmetric node. Hence, for any two nodes like m^l and m_i following uniform distribution, we can say the transmitting or listening state probability of a miner $p^{m^l}_{\tau} = p^{m_i}_{\tau} = p_{\tau} = \frac{1}{\bar{N}+1}$, where expected number of nearby CCNs $\bar{N}_{m^l} = \bar{N}_{m_i} = \bar{N}$, since the task of each miner is uniformly distributed. These preliminaries are used to derive the optimal transmitting or listening state probability of m^l while discovering nearby nodes, which is precisely generated in Algorithm 1, Line 1. N is referred as the time threshold/latency bound, stated in Line 2. Line 3 models the decision vector whether to transmit or listen. In line 4, when $\varepsilon < p_{\tau}$, the node discovery application decides to transmit message including m^l's information & *id* through the communication channel. Else, in line 7, $\varepsilon \not< p_{\tau}$, the node discovery application decides to listen through the communication channel, decode the message, and save *id*. Line 9 updates timetreshold or latency, every time a new m_i is enlisted for task offloading.

Algorithm 1 Optimal Nearby Node Discovery Algorithm for m^l

```
1  Set transmission probability of m^l,
     p^{m^l}_τ : = 1/(N_{m^l}+1), τ : = 0;
2  while τ ≤ N do
3      Generate number randomly, ε ∈ (0, 1) ;
4      if ε < p_τ then
5          Transmission request containing m^l's id and
           associated information via communication
           channel;
6      else
7          Listen through the channel. Successful reception
           is followed by message decoding & saving m_i's
           id;
8      end
9      τ : = τ + 1
10 end
```

Proof: See [43].

We presume that the number of available miners is enough to share task computation. The mining power can be split based upon the nearby miner's computation capability for a given computation task. Since traditional pool mining requires miners to register into the pool (network), miners need to meet a certain standard set by the pool. In our case, we compare the computational capability of a miner and the computation energy required to execute a certain task for that miner to meet the standard. This is further presented in Algorithm 2 and elaborated in Section IV-D.

IV. COMPUTATIONAL COMPLEXITY

Overall computation complexity depends on amount of task (T) required to be computation. Complexity is proportional to the amount of task. In this experiment, if T is larger than the computational capacity of leader miners, then the $t_i \in T | i = 1, 2, 3, \ldots n$ where n number of virtual miners joined the computation. In case of virtual mining, complexity is calculated in terms of communication, computation, offloading (ol), miner selection, etc.

$$C = T_c^{ol} + T_c^{Computation}$$

$$T_c^{ol} = \sum_{i=0}^{n} t_i^{ol}$$

$$T_c^{Com} = \sum_{i=0}^{n} t_i^{mining}$$

following sections presents details of complexity dependencies and calculations.

A. COMPUTATION OFFLOADING

Although mining work varies from application to application, mining involves the execution of numerous tasks that is common, which certainly cannot be completed by any single CCN device. CO plays a significant role in completing the entire work. Thus, the decision to or not to offload computation may result in solo execution or offloading, expressed as $\sigma(m^l) = \{0, 1\}$.

1) SOLO EXECUTION (Mode 0)

As shown in Figure 2, the entire computation of a given T work is done locally at m^l, given that m^l node became the first one to successfully solve the mathematical puzzle, and publish f^{PoW} in the pool, while other m_i miners in M join to form a voting consensus. We denote $\sigma(m^l) = \{0\}$ when m^l decides to execute solo execution process. The offloading to M' is not performed either due to unavailability of CCNs or the offloading tasks can be simply handled by themselves. In this case, FV is not required too.

2) FV FOR COMPUTATION OFFLOADING (Mode 1)

CO is a very complex process of pool mining-affected by various factors, such as mobile users' preferences and capabilities, AP availability, connection quality to transmit VNF, and cloud capabilities. We denote offloading mode, which represents $\sigma(m^l) = \{1\}$, when m^l decides to offload T' tasks to M'. CO is required when the fraction of the $\bar{T}$ tasks are computed locally by m^l, derived from the solo execution part. The rest is offloaded to AP_m for individual m_j miners to download associated VNF recommended by m^l and to compute locally, given that $m_j \in M'$.

B. PRELIMINARY METRICS

This section will formulate the elementary metrics, including processing delay time, VNF loading cost, and energy consumption in *mode* 0 and *mode* 1.

1) SOLO COMPUTATION

We only formulate the delay time and energy consumption (CPU used) in this section. Since there is no FV in solo execution, VNF loading cost will not be considered here.

- **Processing Delay**: It is the sum of time (τ) needed to execute(e) and queue(q) while processing t_i task by m^l, as $\tau^D = \tau^{(m^l,e)} + \tau^{(m^l,q)}$, and execution delay in eq 4, and queuing delay in eq 5 are derived separately as follows:

$$\tau^{(m^l,e)} = \frac{D_n \times G_n}{f^{m^l}} \tag{4}$$

$$\tau^{(m^l,q)} = \frac{Q_n}{f^{m^l}} \tag{5}$$

  where Q_n, is the number of CPU cycles to be executed in the task buffer at m^l such that $q = \{1, \ldots, n\}$.
- **Energy Consumption:** Although m^l tries to complete full work here, it may be available to complete partial work only. In this case, the rest of the work will be offloaded in *mode* 1. Total energy consumption to process T full work and CPU usage:

$$E^{m^l} = \sum_{i=0,\ldots,n}^{t_i} T \left\{ \kappa^{m^l} (f^{m^l})^3 \, \tau^{(m^l,e)} + P_s \tau^{(m^l,q)} \right\} \tag{6}$$

where κ^{m^l} is the computational energy coefficient of the processor's chip of miner m^l [22], [44], and P_s is the static circuit power.

2) REMOTE COMPUTATION

It includes task CO to a group of D2D miners at *mode* 1. Since offloading requires data transmission to configure VNF, total cost is formulated by energy consumption (CPU usage), VNF loading cost, and the processing delay.

- **Processing Delay:** It is different from self computation at *mode* 0, where every tasks are implemented with corresponding VNF and transmitted among individual pool miners (i.e. m_j) for mining. Every pool miner is competing to complete individual task. But this sometimes cause to delay the aggregated time to complete the offloading tasks. Considering the full work T divided into t_n tasks, of which T' tasks are offloaded to multiple m_j miners, until completed at *mode* 1. Since offloading is done simultaneously, the total delay to process T' parts are:

$$\tau^{D_1} = \max_{V_{SFC}=v_0,\ldots,v_{n+1}} \rho[m^l] \times [VNF^{D_1}_{trans} + VNF^{D_1}_{exec} + l^{D_1}_{VNF}] \tag{7}$$

  where $VNF^{D_1}_{trans}$ denotes delay to transmit V_{SFC} to M', while $l^{D_1}_{VNF}$ denotes delay (i.e. queue) in task buffer of AP_m, and $VNF^{D_1}_{exec}$ denotes delay to execute V_{SFC} in M', all in respect to process T' parts implemented with V_{SFC} VNFs for all miner.
- **Transmission delay:** It is the time taken by m^l to transmit the data packets to AP_m.

$$VNF^{D_1}_{trans} = \frac{Data\ size}{transmission\ rate}$$

- **VNF load delay:** It is the waiting time of the task buffer in the processor chip of AP_m while uploading VNFs.

$$l^{D_1}_{VNF} = \frac{Q_n}{f^{AP_m}} \tag{8}$$

  where Q_n is the number of CPU cycles to be executed in the task buffer at AP_m to transmit $v_1, v_3, \ldots, v_n$ into M'.
- **VNF Execution time:** The execution (e) time for M' to compute $v_1, v_3, \ldots, v_n$ VNFs corresponding to T' tasks until completed in *mode* 1:

$$VNF^{D_1}_{exec} = \frac{D_n \times G_{T'} \times T'}{f^{m_x}} \tag{9}$$

  where $G_{T'}$ denotes the computational capability to execute T' task.

- **Energy Consumption:** Sum of the energy consumption of remote processing (*i.e.* E^{m_x}) required to hash T' tasks (i.e. $t_1, t_3, \ldots, t_x$) are as follows:

$$E^{m_x} = \sum_{t_i|i=1,3,\ldots,x}^{T'} \sigma\left[m^l\right] \times \frac{1}{T'} \times \left(P_u \times VNF_{trans}^{D_1} + P_{AP} \times l_{VNF}^{D_1}\right) + \sigma\left[m^l\right]\left[\kappa^{m_x}(f^{m_x})^3 VNF_{exec}^{D_1}\right] \quad (10)$$

where κ^{m_x} is the aggregated computational energy coefficient of the processor's chips of a set of miners M' [22], [44]. P_u and P_{AP} is the transmit power of m^l & AP_m, respectively.

C. PROBLEM FORMULATION WITH CONSTRAINTS

This section studies the optimization of CO scheduling in order to maximize the total net revenue via FV.

1) TOTAL COST OPTIMIZATION

This section elaborates the optimization objectives of CO scheduling:

- **Offloading decision:** Lets presume the net revenue for mining offloadable and non-offloadable tasks be $\Psi\ (m_x)$ and $\Lambda(m^l)$, respectively. Recalling again that offloadable and non-offloadable parts need to hash T' and $\bar{T}$ tasks, respectively.

$$\Psi\ (m_x) = \left[1 - \sigma(m^l)\right]\left[\Upsilon^{m_x} - \lambda_e E^{m_x}\right]$$
$$\Lambda(m^l) = \left[\sigma(m^l)\right]\left[\Upsilon^{m^l} - \lambda_e E^{m^l}\right]$$

- **Total Net Revenue**: The total net revenue for mining the entire work T is:

$$\Xi = \sum_{m_i|i=1,3\ldots,x}^{m_x} \sum_{m_i|i=1}^{m^l} \left[\Psi(m_x) + \Lambda(m^l)\right]$$

2) PROBLEM FORMULATION

Considering the offloading scheduling with decision vectors σ, we in this subsection draw formula for the total optimization problem:

$$\text{Problem 1: } \max_{(\sigma)} \sum_{i=1,3,\ldots,x}^{m_i} \sum_{i=1}^{m^l} \left[\Psi\ (m_x) + \Lambda(m^l)\right]$$

$$\textbf{\textit{s.t.}}\ \text{C1:}\left\{[1 - \sigma(m^l)] + \sigma(m^l)\right\} \sum_{i=1,\ldots,n}^{t_i} T = 1.$$

$$\text{C2:}[1 - \sigma(m^l)] \sum_{i=1,\ldots,n}^{v_i} V_{trans}^{D_1} \times P_{AP} \leq \Omega_{AP}.$$

$$\text{C3:}[1 - \sigma(m^l)] \sum_{m_i|i=1,3,\ldots,x}^{m_x} E^{m_x} \leq f^{m_x}.$$

$$\text{C4:}\sigma(m^l)\tau^D + \left[1 - \sigma(m^l)\right]\tau^{D_1} \leq \tau_n.$$

$$\text{C5:}\frac{D^n}{\bar{T}+T'} \leq L_{VNF}.$$

$$\text{C6:}\boldsymbol{X}_n\left(E^{m^l} + E^{m_x}\right) \leq C_{store}.$$

There are constraints formulated from C1-C6. C1 validates the non-offloading and offloading decision. C2 ensure that the sum data rate of all the miners associated with AP_m doesn't exceed its backhaul capacity Ω_{AP}. C3 confirms that the sum computational capability of all the miners associated with M' does not exceed the sum computational capacity required to execute T' parts. C4 ensures that the task delay does not cross the limit of task deadline τ_n. C5 means that the data size of each offloaded part via D2D link does not cross the limit of the link capacity L_{VNF}. Also, we formulated constraint C6 to ensure that the total size of the data processed in m^l does not exceed its storage capability C_{store}. And $\boldsymbol{X}_n$ refers to the size of the cryptographic hashes of blocks w.r.t. t_n parts, are set at 1 for convenience.

D. TASK OFFLOADING SOLUTION

Overall task offloading processes are executed in two steps, such as 1) Solo mining (Mode 0) and 2) Virtual mining (Mode 1). Virtual mining process is executed only when solo miner/leader is incapable to execute the etire task.

1) EXECUTION IN LEADER MINER (Mode 0)

First function of Algorithm 2 implements solo mining defined as *doMining* function. While, m^l also an m_i, compares it's own computation capability f^{m^l} (stored in I) with the computation energy required G_T (stored in J), in order to complete full work $H(T)$. If its capacity matches then proceed task execution else decide to offload tasks. To implement task execution, the system initially takes input of target value $H(T)$ derived from T, and verifies through while loop in *doMining* function, whether evoked T is already added to BC or not. If not, m^l randomly takes a corresponding input of T value, and keeps in *num*, stated in line 6. In line 8, it adds combination of nonce constantly until rehashing matches the target value $H(T)$, and stores in $\acute{H}_T$. Line , as now $\acute{H}_T$ and input target value $H(T)$ already matched, the *if condition* further calculates credit point (CP) of m^l, and returns f^{PoW} stated in line 10. Else, the loop is repeated in case of not fulfilling the requirements, stated in line 13. Figure 4 (a) reflects the solo execution showing full work $H(T)$ consists of series of tasks, say, $t_1, \ldots, t_5$ which are locally executed because m^l is available and capable. There is no need to offload computation, and share reward. Entire reward is taken by itself only.

2) VIRTUAL MINING (Mode 1)

Mode 1 signifies the CO decision. An important aspect in the CO and reward (or credit point) calculation depends in application model/type, since it determines which fraction of the full work to be processed locally and which are to be processed remotely, what could be offloaded, and how. With the help of Mobile Edge Server Computing station (MESC)/ MANO m_1, m_3, m_4 process offloadable tasks t_1, t_3, t_4 with corresponding v_1, v_3, v_4, respectively, and m^l processes

Algorithm 2 Execution in Leader (m^l) Miner for T Task in n^{th} *mine*

Input : $\langle H(T), T \rangle$
Output: $\langle H(T) \rangle$

1 $I = f^{m^l}$ ▷ m^l's CPU cycles/sec;
2 $J = G_T$, CPU cycles/sec required to reach $H(T)$;
3 **if** $J \leq I$ **then**
4 **Function** `doMining(!`$H(T)$`)`**:**
5 Initiate function *doMining*;
6 $num \leftarrow Random(number)$;
7 $\acute{H}_T \leftarrow H(num)$;
8 **if** $\acute{H}_T \cong H(T)$ **then**
9 $Reward^{m^l} + \leftarrow CP$;
10 **return**
 $Reward^{m^l}, p^{PoW} | f^{PoW} \leftarrow (number, \acute{H}_T)$;
11 **end**
12 **else**
13 Repeat Line 8;
14 **end**
15 **return**
16 **end**
17 **Function** `TenderProcess` $(T', Reward^{T'})$**:**
18 Initiate function *TenderProcess*;
19 $(m_x \subset m_n,\ f^{m_x}) \leftarrow MinerSearch(T', Reward^{T'})$;
20 $(t_1, t_3, \ldots, t_x) \leftarrow parse.Task(T')$;
21 $\{(t_1, m_1), (t_3, m_3), \ldots, (t_x, m_x)\} \rightarrow \{m_1, m_3, .., m_x\}$;
22 **return**
23 $Reward^{m^l} + \leftarrow CP$;
24 **Function** `RewardCalculator()`**:**
25 Initiate function *RewardCalculator*
 $k, p^{PoW} \leftarrow VirtualMining(T', m_j)$;
26 **if** p^{PoW} **then**
27 $Reward^{m_j} + \leftarrow k \times CP$;
28 **end**
29 **return**
30 $Reward^{m_j} + \leftarrow CP$;

Algorithm 3 Virtual Execution of T' in m_j Miner

Input : $H(T')$
Output: $\langle H(T'), \tau_n \rangle$

1 $I^{m_x} = f^{m_x}$ ▷ sum of m_j's CPU cycles/sec;
2 $J^{m_x} = G_{T'}$, CPU cycles/sec required to reach $H(T')$;
3 **Function** `VirtualMining()`**:**
4 Initiate hashing with constant $k \leftarrow 0$;
5 **while** $J^{m_x} \leq I^{m_x}$, $\acute{H}_{T'} \cong H(T')$ **do**
6 $num \leftarrow Random(number)$;
7 $\acute{H}_{T'} \leftarrow H(num)$;
8 **if** $\acute{H}_{T'} \cong H(T')$, $time_{deadline} \leq \tau_n$ **then**
9 $k, p^{PoW} \leftarrow (number, \acute{H}_{T'})$;
10 **end**
11 **else**
12 Repeat Line 8
13 **end**
14 **end**
15 **return**

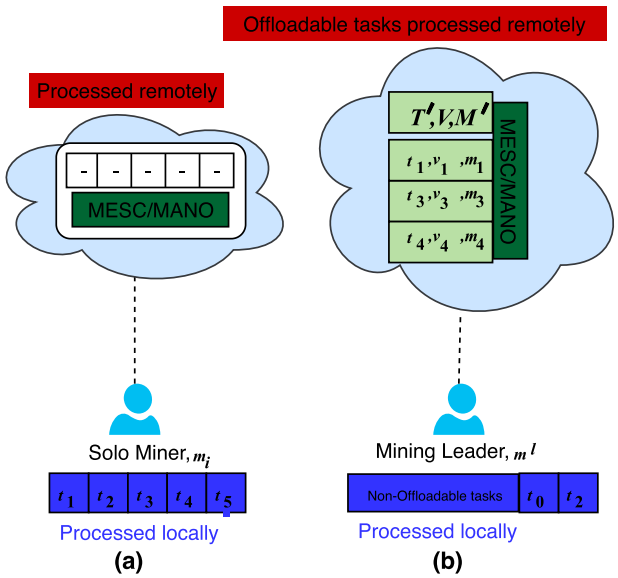


FIGURE 4. Offloading parts in m_x.

non-offloadable t_0, t_2 tasks locally, as shown in Figure 4(b). The application criteria of CO is summarized below.

3) TENDENCY OF APPLICATION OFFLOADABILITY

Offloading is based upon its enabling code, representing how much task is locally processed and how much is off loadable, given that offloading task must be processed in parallel to form a consensus. Selected M' miners are only allowed to process $H(T')$ tasks offloadable by m^l. Initially, Algorithm 2, line 17-21 explains the function *TenderProcess*. This function selects the computation offloadable tasks & miners from the interested enlisted miners from Algorithm 1. Algorithm 2, line 23 determines the portion of reward achieved by m^l while task sharing with f^{m_x}. Line 19 searches for valid miners by verifying their computational capability. Lines 20 parses tasks to a set of selected m_x miners and embeds tasks to respective miners in line 21. Reward of individual m_j is calculated using *RewardCalculator* function, line 24-28. We denote m_j as a virtual miner, shown in Algorithm 2, line 25, as they are now part of the virtual pool. Against every p^{PoW}, there is a constant k value determined, retrieved from Algorithm 3. k represents the workload done by individual m_j through computing its embedded task. Line 30 returns the individual reward of m_j by multiplying k with the credit point (CP).

Virtual execution of VNF in m_j miner is shown in Algorithm 3. Initially, it is presumed that the work done by m_j is zero. The *VirtualMining* function takes T' tasks as input,

TABLE 2. Parameters used to formulate equations.

Parameters	Symbol	Value
Transmit power of miner	P_u	0.1 W
Static circuit power of miner	P_S	0.05 W
Path loss exponent	α	4
Bandwidth/Transmission rate	B	15 Mbps
The power of the noise	ϱ^2	-174 dBm/Hz
Input data size	P_u	30 MB
Delay threshold/time constrain	τ_n	15 s
Computation workload/intensity	G_n	18000 CPU cycles/s
Computation capability of miners	f^{m^l} or f^{m_j}	100-1000 CPU cycles/s
Credit Point	CP	0.1 Token/CP

then compares the processing capacity of m_j miner. The while loop verifies if the evoked T' tasks are already solved or not. If not, m_j randomly takes a corresponding input of T' value, and keeps in *num*, referred in line 6. Then it adds combination of nonce constantly until rehashing matches the target value $H(T')$, and store in $\acute{H}_{T'}$ against a constant k value, stated in line 7. This hashing process should be completed within a given time threshold τ_n, else repeat line 8. Whilst the target value matching occurs, *Reward* of m_j in the *RewardCalculator* function in Algorithm 2 is updated.

V. RESULTS ANALYSIS

This section presents the critical analysis & performance evaluation of the proposed research. The objective is to enable BC technology for IoT objects allowing secure, efficient CO through FV. There are multiple existing efforts considering CO for IoT objects. Still, the proposed research adds more contribution by utilizing BC to ensure transaction security and enables virtualization to ensure the efficiency of IoT objects. The simulation considers Python language for emulating performance evaluation. PoW is considered for transaction verification. The m^l randomly takes value against task computation (T) targeting to reach $H(T)$ and then sorts equal offloading tasks to nearby CCN devices and m^l itself. In the simulation, the random value against task computation (i.e. $H(T)$) is taken from 100-200, where each fraction of T task (i.e. t_i) process 30MB of data. Delay threshold and bandwidth are taken 15 seconds and 15 Mbps, respectively. Transmit power (P_u), and static circuit power (P_S) of individual CCN are taken 0.1 W and 0.05 W, respectively.

Algorithm 2 refers to task selection (i.e. *doMining* function) and reward distribution process (i.e. *RewardCalculator* function), and the nearby CCN devices are selected through Algorithm 1. The algorithm is implemented through Python language, presuming that all nearby CCN devices have equal computation capability and the reward is evenly distributed. The set of nearby CCN device selection is limited between 5-10 nodes at a time randomly. The best combination of node selection is based on the collaborative approach of maximum reward output. e.g., four nodes can compute 100 tasks with a total reward of 10 points, or three nodes can compute 100 tasks with a total reward of 8 points, or two nodes can compute 100 tasks with a total reward of 2 points on. The set of 4 CCNs computing 100 tasks with the reward of 10 points is selected. In this case, leader miner (m^l), as one of the CCN among four nodes, also receives the same reward as other m_j miners, as it computes an equal amount of tasks as others. In this case, m^l computes one-fourth of the computation, such as generating $v_0, v_{(n+1)}$, computing v_2, and matching $v_0 \cong v_{(n+1)}$, while individual m_1, m_3, m_4 computes v_1, v_3, v_4 VNFs respectively. It is presumed generating $v_0, v_{(n+1)}$ and matching $v_0 \cong v_{(n+1)}$ requires negligible amount of computation, and so reward for this work is ignored. Parameters and symbols used to implement proposed algorithms are stated in Table 2.

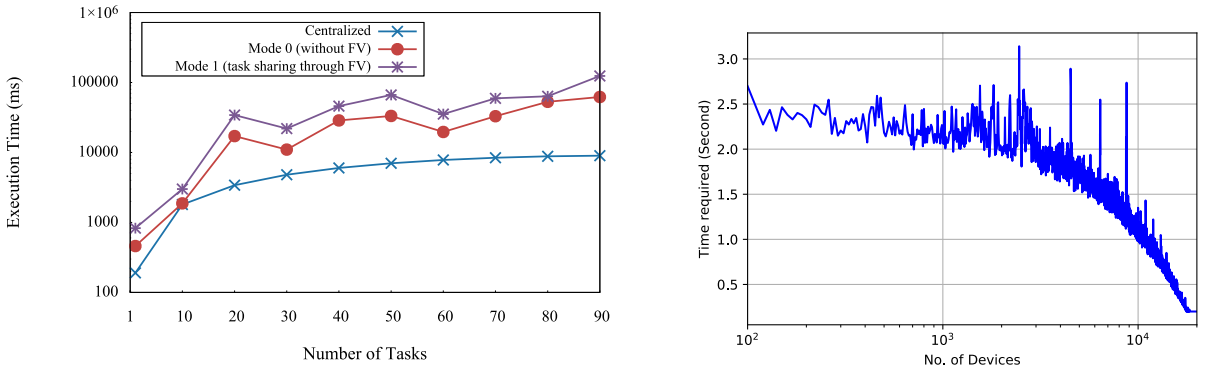
A. EVALUATION AND DISCUSSION

Function virtualization impact on mining has been evaluated with state-of-art in Figure 5. Figure 5a shows comparison between traditional centralized approach and decentralized BC approach with or without FV for task computation. It presents the execution time (milliseconds) of compute tasks. IoT nodes are resource constrained devices that are not capable of computing heavy tasks. In Figure 5a, the blue line refers to a server node computing task via a centralized approach that considers no FV or BC technology, the red line refers to computing tasks by a CCN using BC platform, the purple line refers to CO to nearby CCNs by a m^l via FV using BC platform.

The time threshold (τ_n) is set as 15 seconds. For example, to compute ten tasks, a server node consumes 1800 ms in a centralized approach, a CCN consumes 1870 ms to compute the same task in *Mode* 0, and an m^l in *Mode* 1 consumes 2992 ms to share task computation with the selected nearby CCN miners to complete the same tasks. It is noticeable that the centralized approach consumes the least amount of time, and *Mode* 1 takes the maximum amount of time to execute a certain amount of tasks. *Mode* 0 is moderately taking less time than *Mode* 1 because the computation capability (i.e., $f^{m^l} =$ 18000 CPU cycles/sec) of m^l is just enough to compute the given task by itself. As the task increases, the execution time increases steadily in a centralized approach, while the decentralized approach consumes more time in *Mode* 0 and *Mode* 1. Successful block creation consumes more time because the difficulty for block transaction verification increases proportionately as more CCNs participate in the mining process.

Figure 5b presents a computation time with the participation of miner devices for mining a specific task through FV. It is observed that time is disproportionate to the number of active participants. While the number of FV participants is increased, computation time is decreased. It is assumed that every device has a 1k to 50k T computation capacity. The task is split randomly among devices. It can vary based on the devices' capacity.

This ensures the security of IoT nodes. Even though transmission delay of CO is minimized through FV, internet



(a) Impact of FV

(b) Virtual Collaborative Mining of One Task

FIGURE 5. Traditional centralized approach and BC based decentralized approach with and without FV for task computation.

bandwidth (B), presumed as 15 Mbps, remains a crucial factor in minimizing computation execution time. It is noticeable that there is a trade-off between time and energy consumption while incorporating a decentralized approach. The centralized approach consumes less time but is limited to provisioning security. In comparison, the decentralized BC approach with FV collaboratively consumes a little more time but provides more security and resource optimization by employing underutilized nearby CCNs. But individual CCN takes only a couple of 100ms to execute each virtual functions, which is a lot less than the centralized approach. The proposed research considers FV to offload computation tasks to nearby CCNs while m^l is not computed for all tasks by itself due to its limited computation capability. The transaction security of CCNs are preserved by BC and FV ensures efficient CO to nearby devices.

Figure 6 shows computation usage of participating miners via FV using the BC platform. It presents the number of tasks to be computed by nearby participated CCNs and their average computation (CPU cycles/sec). The blue bars refer to the number of CCNs participating in computation mining. The orange bar refers to the average computation of each set of CCN miners, e.g., four CCNs participates in computing 100 tasks with an average computation of 6.89 CPU cycles/sec, or 3 CCNs participates to compute 110 tasks with an average computation of 6.62 CPU cycles/sec, and so on. It is noticeable that the number of participating CCN miners varies, affecting average computation to compute every set of tasks. As the difficulty of block mining is connected to the number of participating miners, some tasks require more energy and time to compute. In comparison, others require less energy and time to compute. Transmission delay of virtual functions also affects CO, as tasks remain in the queue to be offloaded to the nearby CCNs.

Figure 7 shows the average reward distribution for the participating miners for task computation. The x-axis refers to the number of tasks, and the y-axis refers to the average reward points (CP) for each set of participating CCN miners for a set of given tasks. For example, four CCNs participate

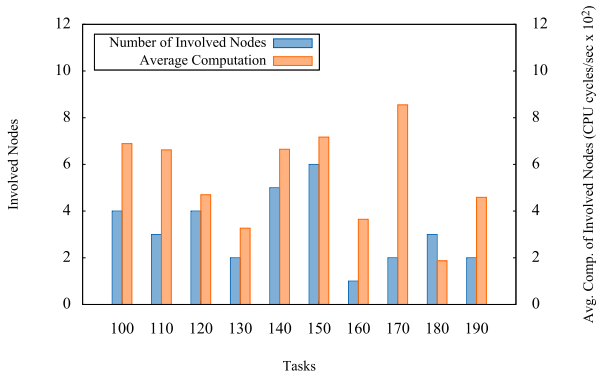


FIGURE 6. Computation usage of participating miners via function virtualization.

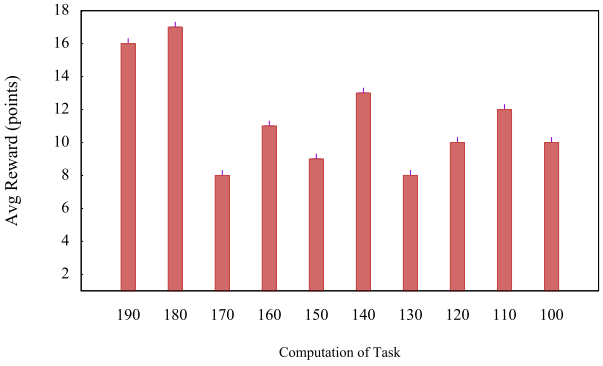


FIGURE 7. Reward distribution for computation offloading.

in computing 100 tasks to earn an average reward point of 10, or 2 CCNs participate in computing 130 tasks to earn an average reward point of 8, and so on. It is noticeable that the obtained CP is influenced by the number of miners participating in the computation mining and the related number of tasks that need to be computed by each set of participating miners. Among other factors, bandwidth tends to be very crucial.

Figure 8 shows individual reward (CP) and computation usage of participating miners via FV using the BC platform.

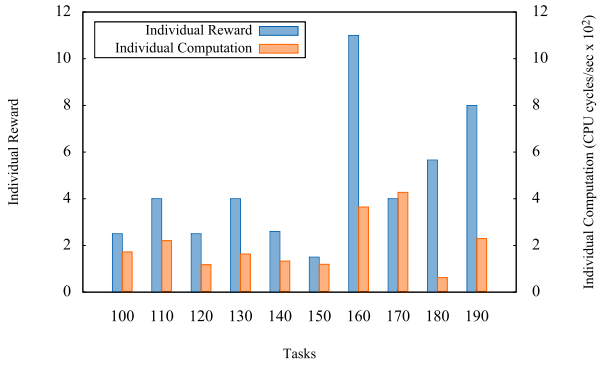


FIGURE 8. Individual reward & computation of participating miners via function virtualization.

The x-axis refers to the number of tasks to be computed by nearby CCNs, and the y-axis refers to individual rewards (CP) of CCN participating in computation mining. The right of the y-axis refers to the individual computation (CPU cycles/sec) of the involved CCN for executing the single task. The blue bars refer to individual rewards (CP), and the orange bar refers to the individual computation of executing a single task of participating miners. e.g., four CCNs participate in computing 100 tasks to earn an individual reward point of 2.5, or three CCNs participate in computing 110 tasks to earn an individual reward point of 4, and so on. It is presumed participating CCN miners have equal computation capabilities where $f^{m_j} =$ 100-1000 CPU cycles/sec. So the shared task computation is evenly distributed as it would take the same time to execute a task embedded with a virtual function. As a result, the rewards are also evenly distributed. Internet bandwidth (B) remains 15 Mbps. And the time taken by individual m_j to execute each VF (v_i) is also equal. It is worth mentioning that FV enables efficient processing of CO. Reward distribution for block mining is validated by cryptographic hashing functions, where the difficulty starts with '0000'. It also creates a chain of linked blocks which ensures data integrity and the immutability of the ledger.

- **Insights:** From the above discussion, it is predictable that FV enables optimal resource usage and allows heavy computation mining for resource-constrained IoT devices utilizing the virtual task mining process. As the underutilized IoT devices are used to compute mining tasks, the cost of installing expensive server stations is no longer required. The net revenue of individual CCN is optimized as it can earn more CPs rather than being underutilized. It is also noticed that hashing also leverages the PoW to validate transactions encouraging incentive miners to agree upon computation mining.

VI. COMPLEXITY OF COMPUTATION OFFLOADING

CO application may require individual miners to consider several internal components to exercise to improve real-time CO efficiency. They are (i) Program/code profiler component, (ii) System profiler component, and (iii) Decision engine component [45]. The first one is responsible for determining

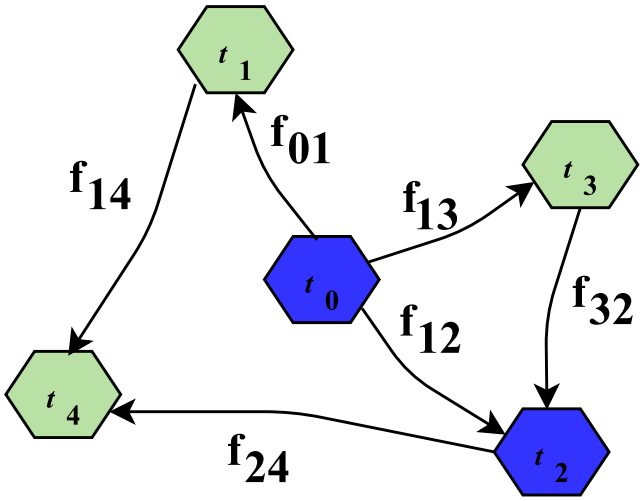


FIGURE 9. Dependency graph of Offloadable tasks.

the offload table tasks based on application decision type and selecting which tasks need to be processed in m^l and which to be processed in m_j. The second is responsible for configuring available bandwidth validate miners, and CP distribution is based on energy spent locally & remotely. Finally, the last one decides to or not to offload computation. Moreover, some critical discussions about CO are stated below.

- **Knowledge based data processing:** m^l initially compares between the amount of data that is required to process the entire work and the amount of data it can locally process, shown in Algorithm 2, line 3. This comparison already shows how much data full work consumes (stated in problem formulation $C1$). Based on this assumption, it is also calculated how much time the entire work requires to complete (stated in problem formulation $C4$). Most importantly, what fraction of the computation needs to be offloaded for continuous application processing (stated in problem formulation $C2$). We next explain the complexity of the inter-dependency offloading issues through Figure 9.
- **Inter-dependency of the offloadable tasks:** The application decision regarding CO is determined by the dependency relationship of T' tasks to be processed. However, offloading tasks may or may not be dependent on each other. Independent tasks may be processed locally in m^l, based on application decision. In the case of dependent tasks, the continuation of application execution may require the input of one task from the output of another task (s), whom we say are mutually dependent on each other. Hence, parallel offloading may not always be possible. We further demonstrate this complexity through a Component dependency Graph (CDG) [46]–[49] shown in Figure 9. The entire application is broken into two parts; (i) Offloadable tasks (i.e. $t_1, t_3,$ *and* t_4), and (ii) Non-Offloadable tasks (i.e. t_0, *and* t_2). Note that t_1 and t_3 are only offloadable after processing t_0, while t_4 is only offloadable

after processing t_0, t_1, t_2, t_3. This means that certain tasks are only offloadable after completion of dependent non-offloadable tasks, which are processed locally. We intend to mitigate this issue in the future.

VII. CONCLUSION

This effort aimed to investigate existing challenges of traditional IoT ecosystems and find the optimal solutions for challenges integrating SDN and NFV with BC technology to mitigate the storage and computation issue of resource-constrained IoT devices. We have discussed significant research that exists regarding BC and FV. The literature review separately grouped existing efforts in two parts, (i) existing BC mining strategies, (ii) FV strategies for CO. Although the review section has separately addressed many ways to mitigate resource-constrained IoT devices' storage and computation challenges, it was not possible to achieve the desired goal. As most IoT devices are limited in computational capability and storage space, a single IoT device cannot do extensive computation. We then further extended our effort to produce a contribution focusing on BC mining designed explicitly for resource-constrained IoT devices. Our goal to this end was to render underutilized resource-constrained IoT devices for task computation with minimal energy consumption within the given time constraint. We have combined BC and FV to mitigate the said issue. In light of this, we introduced a novel BC framework for such mobile IoT devices utilizing NFV and SFC technologies on top of SDN controllers. The contribution adopts a virtual mining strategy that adheres to the computation-intensive PoW puzzle performed by the individual nearby mobile nodes (or CCNs). The CO mechanism is performed by a lead miner, which coordinates a series of VNFs linking them through SFC technology. The leader miner eventually accumulated the execution result of multiple VNFs to find the full PoW. We have shown that FV can reduce computation cost, minimize energy consumption, and make block mining more efficient than a centralized approach.

REFERENCES

[1] I. Eyal, "The miner's dilemma," in *Proc. IEEE Symp. Secur. Privacy (SP)*, Washington, DC, USA, May 2015, pp. 89–103, doi: 10.1109/SP.2015.13.

[2] *Cryptocompare*, 2019. Accessed: Apr. 24, 2019.

[3] T. Xue, Y. Yuan, Z. Ahmed, K. Moniz, G. Cao, and C. Wang, "Proof of contribution: A modification of proof of work to increase mining efficiency," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Jul. 2018, pp. 636–644.

[4] *Internet of Things (IoT) Connected Devices Installed Base Worldwide From 2015 to 2025*. (in Billions). 2019. Accessed: Apr. 4, 2019.

[5] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[6] O. Kaiwartya, A. H. Abdullah, Y. Cao, J. Lloret, S. Kumar, R. R. Shah, M. Prasad, and S. Prakash, "Virtualization in wireless sensor networks: Fault tolerant embedding for Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 571–580, Apr. 2018.

[7] F. R. Yu, J. Liu, Y. He, P. Si, and Y. Zhang, "Virtualization for distributed ledger technology (vDLT)," *IEEE Access*, vol. 6, pp. 25019–25028, 2018.

[8] G. Lee, M. Kim, S. Choo, S. Pack, and Y. Kim, "Optimal flow distribution in service function chaining," in *Proc. 10th Int. Conf. Future Internet*, New York, NY, USA, Jun. 2015, pp. 17–20, doi: 10.1145/2775088.2775103.

[9] H. Jin, X. Zhu, and C. Zhao, "Computation offloading optimization based on probabilistic SFC for mobile online gaming in heterogeneous network," *IEEE Access*, vol. 7, pp. 52168–52180, 2019.

[10] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[11] D. Zhao, D. Liao, G. Sun, and S. Xu, "Towards resource-efficient service function chain deployment in cloud-fog computing," *IEEE Access*, vol. 6, pp. 66754–66766, 2018.

[12] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho, "Application component placement in NFV-based hybrid cloud/fog systems with mobile fog nodes," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1130–1143, May 2019.

[13] M. D. Ananth and R. Sharma, "Cloud management using network function virtualization to reduce CAPEX and OPEX," in *Proc. 8th Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Dec. 2016, pp. 43–47.

[14] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 518–532, Sep. 2016.

[15] N. Houy, "The bitcoin mining game," Tech. Rep., 2014. Accessed: Jul. 6, 2019.

[16] R. P. B. A. Fisch and A. Shelat, "The bitcoin mining game: On the optimality of honesty in proof-of-work consensus mechanism," Tech. Rep., 2017. Accessed: Jul. 6, 2019.

[17] B. A. Fisch, R. Pass, and A. Shelat, "Socially optimal mining pools," 2017, *arXiv:1703.03846*. Accessed: Jul. 6, 2019.

[18] L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor, "On power splitting games in distributed computation: The case of bitcoin pooled mining," in *Proc. IEEE 28th Comput. Secur. Found. Symp.*, Jul. 2015, pp. 397–411.

[19] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[20] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2017, pp. 1–6.

[21] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and CPU time allocation for mobile edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[22] S. Bi and Y. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[23] J. Kang, O. Simeone, and J. Kang, "On the trade-off between computational load and reliability for network function virtualization," *IEEE Commun. Lett.*, vol. 21, no. 8, pp. 1767–1770, Aug. 2017.

[24] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.

[25] S. Biswas, K. Sharif, F. Li, B. Nour, and Y. Wang, "A scalable blockchain framework for secure transactions in IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4650–4659, Jun. 2019.

[26] Z. Wu, Z. Shi, and Y. Zeng, "SFC orchestration method based on energy saving and time delay optimization," in *Proc. 9th Int. Conf. Comput. Eng. Netw.*, Jul. 2020, pp. 347–356.

[27] Y. Kano and T. Nakajima, "An alternative approach to blockchain mining work for making blockchain technologies fit to ubiquitous and mobile computing environments," in *Proc. 10th Int. Conf. Mobile Comput. Ubiquitous Netw. (ICMU)*, Oct. 2017, pp. 1–4.

[28] F. Lin, Z. Zheng, Z. Huang, C. Tang, H. Peng, and Z. Chen, "A sustainable reward mechanism for block mining in PoW-based blockchain," in *Proc. 5th Int. Conf. Inf., Cybern., Comput. Social Syst. (ICCSS)*, Aug. 2018, pp. 156–161.

[29] A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis, "Blockchain mining games," in *Proc. ACM Conf. Econ. Comput. (EC)*, New York, NY, USA, Jul. 2016, pp. 365–382, doi: 10.1145/2940716.2940773.

[30] J. Bae and H. Lim, "Random mining group selection to prevent 51% attacks on bitcoin," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2018, pp. 81–82.

[31] R. Abe, K. Nakamura, K. Teramoto, and M. Takahashi, "Attack incentive and security of exchanging tokens on proof-of-work blockchain," in *Proc. Asian Internet Eng. Conf. (AINTEC)*, New York, NY, USA, Nov. 2018, pp. 32–37, doi: 10.1145/3289166.3289171.
[32] S. Biswas, K. Sharif, F. Li, S. Maharjan, S. P. Mohanty, and Y. Wang, "PoBT: A lightweight consensus algorithm for scalable IoT business blockchain," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2343–2355, Mar. 2020.
[33] G. Sun, M. Dai, J. Sun, and H. Yu, "Voting-based decentralized consensus design for improving the efficiency and security of consortium blockchain," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6257–6272, Apr. 2021.
[34] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.
[35] Y. Cheng, L. Sun, and X. Liu, "User-oriented energy-saving offloading for wireless virtualization aided mobile edge computing," in *Proc. 2nd Int. Conf. Comput. Sci. Appl. Eng. (CSAE)*, New York, NY, USA, 2018, pp. 51:1–51:5., doi: 10.1145/3207677.3277931.
[36] Y. Cheng, P. Zhao, and L. Wang, "User-oriented green computation in small cell networks with mobile edge computing," in *Proc. 4th Int. Conf. Commun. Inf. Process. (ICCIP)*, New York, NY, USA, 2018, pp. 246–250, doi: 10.1145/3290420.3290456.
[37] Y. Cheng, L. Yang, and H. Zhu, "Distributed computation offloading and power allocation for wireless virtualization aided mobile edge computing," in *Proc. 2nd Int. Conf. Comput. Sci. Appl. Eng. (CSAE)*, New York, NY, USA, 2018, pp. 55:1–55:5, doi: 10.1145/3207677.3277938.
[38] H. Zannat and M. S. Hossain, "A hybrid framework using Markov decision process for mobile code offloading," in *Proc. 19th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2016, pp. 31–35.
[39] B. Hibbard, "Model-based utility functions," *J. Artif. Gen. Intell.*, vol. 3, no. 1, pp. 1–24, Jan. 2012, doi: 10.2478/v10229-011-0013-5.
[40] C. Sun, J. Zhou, J. Liuliang, J. Zhang, X. Zhang, and W. Wang, "Computation offloading with virtual resources management in mobile edge networks," in *Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring)*, Jun. 2018, pp. 1–5.
[41] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.
[42] N. I. M. Enzai and M. Tang, "A heuristic algorithm for multi-site computation offloading in mobile cloud computing," *Proc. Comput. Sci.*, vol. 80, pp. 1232–1241, Jun. 2016, doi: 10.1016/j.procs.2016.05.490.
[43] T. Shen, Y. Wang, Z. Gu, D. Li, Z. Cao, H. Cui, and F. C. M. Lau, "Alano: An efficient neighbor discovery algorithm in an energy-restricted large-scale network," in *Proc. IEEE 15th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2018, pp. 353–361.
[44] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.
[45] H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya, "Mobile code offloading: From concept to practice and beyond," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 80–88, Mar. 2015.
[46] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81–93, Jan. 2015.
[47] Y. Zhang, H. Liu, L. Jiao, and X. Fu, "To offload or not to offload: An efficient code partition algorithm for mobile cloud computing," in *Proc. IEEE 1st Int. Conf. Cloud Netw. (CLOUDNET)*, Nov. 2012, pp. 80–86.
[48] M. Deng, H. Tian, and B. Fan, "Fine-granularity based application offloading policy in cloud-enhanced small cell networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, May 2016, pp. 638–643.
[49] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 301–313, Apr. 2019.

JUN WANG received the bachelor's degree in communication engineering and the master's degree in communication and information system from the University of Electronic Science and Technology of China, in June 2005 and June 2008, respectively. He is currently a Senior Engineer with the 30th Research Institute of China Electronic Technology Group Corporation.

JIANG ZHU received the bachelor's degree in computer science and technology from Jiangnan University, in 2013, the master's degree in business administration from Asia Metropolitan University, in 2017, and the Ph.D. degree in educational management from Jose Rizal University, in 2022. He is currently a Teaching Assistant with Shanghai SIPO Polytechnic.

MINGHUI ZHANG received the bachelor's degree in computer and applications from National Defense University, in 2002, and the master's degree in software engineering from the Huazhong University of Science and Technology, in 2010. He is currently a Lecturer with the Guangdong Innovative Technical College.

IQBAL ALAM received the bachelor's degree in business information and communication system from London Metropolitan University, in 2009, and the master's degree in software engineering and the Ph.D. degree in computer science and technology from the Beijing Institute of Technology, in 2016 and 2020, respectively. He is currently a Publication Manager with the Nan Yang Academy of Sciences (NASS).

SUJIT BISWAS (Member, IEEE) received the Ph.D. degree in computer science and technology from the Beijing Institute of Technology, China. He is currently a Research Fellow in blockchain and AI with the University of Surrey and an Honorary Research Associate with the Centre for Blockchain Technologies (CBT), University College London (UCL), U.K. He is also an Assistant Professor with the Faridpur Engineering College, University of Dhaka, Bangladesh. His research interests include the IoT, blockchain, sensor networks, machine learning, federated learning, and deep learning. He is a Life Member of the Bangladesh Computer Society (BCS).