Received: 28 October 2019 | Revised: 22 November 2019 | Accepted: 2 December 2019

DOI: 10.1002/spe.2786

SPECIAL ISSUE PAPER

WILEY

A mobile data offloading framework based on a combination of blockchain and virtual voting

Vikas Hassija[1] | **Vikas Saxena**[1] | **Vinay Chamola**[2]

[1]Department of CSE and IT, JIIT, Noida, India

[2]Department of Electrical and Electronics Engineering, BITS-Pilani, Pilani, India

Correspondence
Vinay Chamola, Department of Electrical and Electronics Engineering, BITS-Pilani, Pilani, India.
Email:
vinay.chamola@pilani.bits-pilani.ac.in

Summary

The emergence of mobile cloud computing enables mobile users to offload computation tasks to other resource-rich mobile devices to reduce energy consumption and enhance performance. A direct peer-to-peer connection among mobile devices to offload computation tasks can be a highly promising solution to provide a fast mechanism, especially for deadline-sensitive offloading tasks. The generic blockchain-based system might fail in such a scenario due to it being a heavyweight mechanism requiring high power consumption in the mining process. To address these issues, in this article, we propose a directed acyclic graph-enabled mobile offloading (DAGMO) algorithm. DAGMO model is empowered by traditional blockchain features and provides additional advantages to overcome the fundamental limitations of generic blockchain. A game-theoretic approach is used to model the interactions between mobile devices. The numerical analysis proves the proposed model to enhance the overall welfare of the participating nodes in terms of computation cost and time.

KEYWORDS

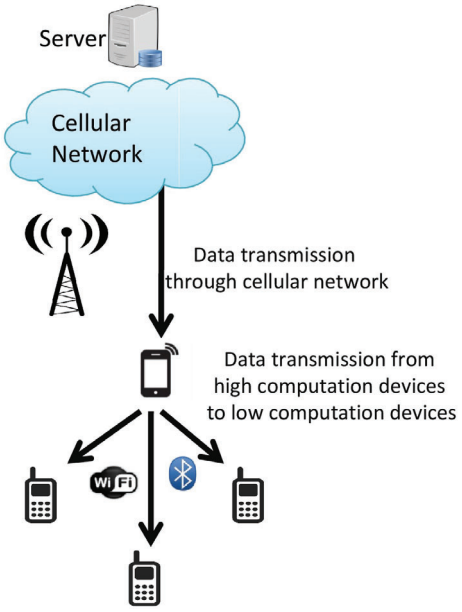
big-data, blockchain, consensus finality, directed acyclic graph, distributed applications, mobile offloading

1 | INTRODUCTION

There is a rapid increase in the number of mobile devices and computation-intensive mobile applications. These applications include video streaming, augmented reality applications, audio/video conferences, collaborative editing, and so on. The focus of these applications is toward enhancing the quality of service (QoS) for the end-users. However, by enhancing the QoS, these applications end up generating a large amount of mobile traffic, thus creating a big challenge for mobile network providers. According to a recent report from Cisco, the rate of mobile data traffic generation is expected to increase from 11.7 exabytes per month in 2018 to 31 exabytes per month by 2020.[1] The mobile data traffic in the world is expected to grow from 56% in 2018 to 80% in 2022.[1] With the rapid increase in the number of mobile devices and mobile data traffic, a stable and fast network connection will be required to maintain the QoS for end-users. However, the network providers are not motivated to continuously expand the network infrastructure to improve the QoS for the users due to heavy investments involved in infrastructure expansion.

Mobile data offloading is one of the most promising solutions that can be used to cope up with these issues. Figure 1 shows a primary use case of resource optimization using mobile data offloading. Mobile data offloading uses different networking techniques to deliver the requested data that would have been provided by the cellular network otherwise.[2,3]

FIGURE 1 A sample use case for mobile data offloading



These techniques end up significantly reducing the operational cost for the network providers and help in enhancing the QoS and quality of experience (QoE) for the end-users. In comparison with the data offloading task, the computation offloading tasks are more delay-sensitive.[4] The computation offloading involves uploading the data from the service requester to the service provider and then downloading the computation results.[5]

Major works related to mobile data offloading include offloading through small cell networks (SCNs), Wi-Fi offloading, offloading through opportunistic, and heterogeneous networks.[6] The SCN approach involves the deployment of small base stations such as micro, pico, or femto base stations to offload the mobile data traffic. Although such techniques can help in reducing the overall mobile data traffic, these techniques are again associated with infrastructure expansion. Apart from infrastructure-related costs, the range of these small base stations is limited from a few meters in urban areas to one or two kilometers in rural areas. Wi-Fi offloading techniques depend on offloading the data transfer and computation task over the Wi-Fi access points (APs).[7] Such APs are limited in number and range. The opportunistic networks depend on the device-to-device (D2D) communication to offload the data.[8] The authors use a store-carry-forward strategy in such systems. Some mobile devices can store the data in the buffer and can carry it when they are moving and can forward it to other mobile users.[9] Although such networks are free from any further infrastructure expansion, it is difficult to predict the mobility pattern in such networks. Also, there is no mechanism to motivate or incentivize mobile devices or users to actively participate in the offloading process. Heterogeneous network offloading is a combination of SCN and opportunistic network offloading technique.[10,11]

In this article, a secure distributed ledger technology (DLT)-based approach is proposed to create a peer-to-peer network of mobile users to offload data and computation tasks. There are very few works in the literature, which use DLT for mobile data offloading.[12,13] Generic blockchain-based algorithms are used in these works to enable a peer-to-peer network of mobile users to perform data and computation offloading. However, these approaches lack scalability and efficiency due to some fundamental constraints in the traditional blockchain architecture. Generic blockchain involves the use of miners to add the blocks in the chain, and they have to be paid by the network. The offloading tasks often include microtransactions between the service providers and users, which are much less than the miner's incentive to add a block in the chain.[14,15] The forking and pruning process in normal blockchain reduces the efficiency of work done by the network nodes. Intuitively, if multiple nodes attempt to provide a video requested by a single user, then the work done by all attempting nodes will be pruned except one. Over and above, the traditional blockchain algorithm does not consider the time-stamp ordering of the transaction entry while processing them. It might not be necessary to order the simple financial transactions, but for the resource-constrained mobile devices, a consensus time-stamp ordering is highly important. Most of the offloading tasks are delay-sensitive, and therefore, it is first important to process the requests that are closer to the deadline.[16] To solve these issues, we propose a directed acyclic graph (DAG)-based peer-to-peer network of mobile users. The proposed model is based on blockchain fundamentals and is, at the same time, highly scalable and

efficient. The proposed model also maintains a perfect consensus time stamp with probability one. More specifically, the major contributions of this work are summarized as follows.

- We create a blockchain-based peer-to-peer network of mobile users where users can securely enter or leave the network and can perform data and computation offloading tasks.
- A hashgraph consensus algorithm is used to schedule the offloading tasks based on the least deadline time for computation.
- A game-theoretic model is presented to negotiate and choose the best mobile device with high computation power to compute tasks of mobile devices with low computation power in a cost and time-optimal way.
- The simulation results demonstrate that our proposed schemes achieve the lowest communication cost and perfect scheduling as compared with other offloading schemes.

The rest of this article is organized as follows. Section 2 presents the recent work related to mobile data offloading. Section 3 explains some background information related to the DAG and order of the tasks using a hashgraph consensus algorithm. Section 4 presents the mobile data offloading algorithm and calculation of computation cost and time. Section 5 presents a game-theoretic model to select the appropriate mobile device in the network for cost-optimal offloading. Section 6 presents the simulation settings and numerical analysis results, and Section 7 finally concludes the article.

2 | RELATED WORK

Mobile data and computation offloading works are rapidly increasing. There have been various attempts in recent years to provide data and computation offloading near to the mobile device. These techniques help in reducing the cost and delay involved in task accomplishment for the user and the overall data traffic on the mobile network. Zhou et al[6] have presented a detailed survey of all the existing methodologies used for performing the data and computation offloading tasks. The techniques are majorly divided into four categories involving SCNs, Wi-Fi networks, opportunistic mobile networks, and heterogeneous networks. Rimal and Maier[17] have discussed a fiber-wireless (Fiwi) integration technique to increase the range of offloading.

Gao et al[18] have discussed mobile data offloading in D2D opportunistic networks. A single user can download a file from the cellular network, and the first user can fulfill a similar request for other users in the vicinity. The deadline constraints are also considered, along with minimizing the monetary cost involved in the offloading tasks. Liu et al[7] present a prediction-based mobile data offloading in mobile cloud computing (MCC). The authors formulate the offloading task as a finite horizon Markov decision process. A hybrid offloading model is used where multiple networks are used to transfer the data. Mehmeti and Spyropoulos[19] have discussed the need and performance improvements by using delayed mobile data offloading. Delayed mobile data offloading is being used by various mobile network providers where the offloading task is delayed until the Wi-Fi network is available. Table 1 summarizes the recent related works in the direction of mobile data offloading with their unique contributions.

Flores et al[20] have proposed a crowd-sensed evidence trace mechanism to optimize the offloading task and to predict the benefits from offloading. Offloading a task to another device or network may sometimes be beneficial and not always. The authors discuss the prediction analysis of the benefits of offloading a task. The authors of References 21 and 23 have discussed the need and benefits of considering the social context, that is, the user's social relationships and the popularity of a mobile application while offloading. The authors of References 22 and 24 have discussed a game-theoretic model to optimize the offloading task pricing in case of overlaps in the range of service providers. A data compression technique to save energy in the offloading task is discussed in References 25 and 26. The offloaded data are first compressed before transmission to reduce the data size and latency. Table 1 summarizes the recent related works in the direction of mobile data offloading with their unique contributions.

Although there have been various recent attempts to perform mobile data and computation offloading tasks, there is no peer-to-peer, secure, and cost-optimal framework existing for the same. The proposed framework gives a significant focus on creating a peer-to-peer network of mobile users, where the users can act as an offloading service provider or a consumer, thereby enhancing the ultimate QoE of the participating users. Users can securely enter or leave the network and can decide on the cost-optimal solution for data and computation offloading. A game-theoretic model is used to

TABLE 1 Related work on mobile data offloading

Year	Author	Contributions
2017	Rimal and Maier[17]	Fiwi integration technique to increase the range of offloading.
2017	Gao et al[18]	Mobile data offloading in D2D opportunistic networks.
2017	Mehmeti et al[19]	Performance improvements by using delayed mobile data offloading.
2018	Zhou et al[6]	Survey of all the existing methodologies for data and computation offloading.
2018	Liu et al[7]	Prediction-based mobile data offloading in MCC.
2018	Flores et al[20]	Crowd-Sensed evidence trace mechanism to optimize the offloading task.
2019	Cheon et al[21]	Benefits of considering the social networking context while offloading.
2019	Li et al[22]	Game-theoretic model to optimize the offloading task pricing in case of overlaps.

Abbreviations: D2D, device to device; MCC, mobile cloud computing; Fiwi, fiber-wireless.

perform offloading tasks in a cost-optimal manner. Consensus time stamp of the offloading requests in the network is maintained, and the tasks are scheduled based on the perfect ordering using the hashgraph consensus algorithm.

3 | PROPOSED MODEL FOR MOBILE DATA AND COMPUTATION OFFLOADING

In this section, the distributed network of mobile users for data and computation offloading is discussed. The data offloading scenario, as shown in Figure 1, is the case where the data are taken from the neighboring devices rather than directly from the cellular network. Another type of offloading scenario is computation offloading, where the nearby high computation devices are used to perform some data computation for low computation devices. The proposed model is generic and can be used both in data offloading and computation offloading scenarios. Using a distributed mobile offloading algorithm, all users with high and low computation power can be part of the same distributed network. The users can securely request an offloading task by submitting a transaction on a DAG, and the users with high computation power can submit the best possible options in terms of cost and time for the offloading task.[27,28] The iterative auctioning model can help in deciding the appropriate price for the task based on the demand-supply ratio.[29] This would help in maximizing the profit of both the users. A smart agent is introduced in the form of a smart contract that does the task of analysis and provides the user with the set of best offloading solution, rather than flooding the user with all the available options.[30] In this section, we discuss various prelims for network creation and consensus calculation.

3.1 | Digital identity

As soon as a user wishes to enter the network, a set of a private and public key is generated for the user using an elliptic curve digital signature application.[31,32] A random number generator is used to create a large random number to be used as the private key d.[33,34] The public key is computed using the random private key using equation $Q(X, y) = d \times G(X, y)$ through point multiplication. Here $Q(x, y)$ acts as the public key of the user, and $G(x, y)$ is the domain parameter.[35] The set of keys is used for all the transactions or communication taking place between different mobile users in the network.[36,37] The issuing party needs to sign the transaction using the private key digitally, and all other intended users can verify the transaction using the public key of the signing authority.[38] This eventually prevents the problem of nonrepudiation.[39,40]

Once the user gets the set of keys, he/she can request a parking lot by filling the basic details, including area, duration of parking, time to reach, and the maximum price that the user is willing to pay.[41] The details mentioned above will be packed in a transaction container and will be flooded in the network of users in the form of a gossip event. The event architecture is presented in Figure 2. The users will "gossip the gossip," and the event will reach all the network nodes exponentially. The gossip protocol is used here for communication as it requires very less bandwidth to gossip the DAG created as compared to sending the signed transaction to all the nodes. The detailed process of parking lot allocation and scheduling using gossip protocol is explained in the next section.

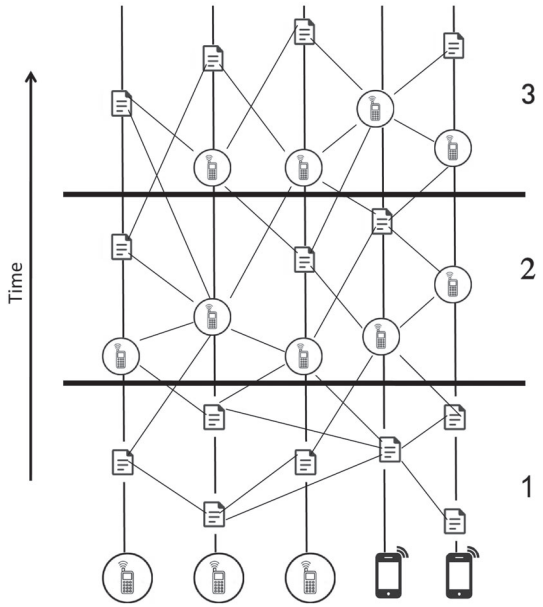


FIGURE 2 Distributed network using hashgraph consensus algorithm

3.2 | Hashgraph consensus algorithm

Hashgraph works on a consensus algorithm. The DAG build to calculate the hashgraph consensus is shown in Figure 2. The main objective of this consensus algorithm is to bring all the nodes to a final agreement on the order of offloading requests.[32] The graph grows in a single direction, and each node shares a copy of the same graph. Hashgraph consensus algorithm helps in achieving the perfect ordering of the requests with consensus finality.[42] Hashgraph helps in achieving a level of trust among all the nodes in the distributed network. The generic hashgraph consensus is used to get the time-stamp order and is further improvised to give the final ordering based on deadline and time stamp.

Hashgraph is asynchronous byzantine fault tolerant. This means that no node in the network can influence other nodes to reach or not reach the consensus. Also, the nodes do not have the authority to change the consensus order once it is reached. The hashgraph algorithm is deterministic in nature, and every node in the network will eventually reach to a consensus or agreement with other nodes.[32,43] Hashgraph is also distributed denial of service resilient. No node or a small group of nodes in the network is given special rights and responsibilities in achieving consensus. This also increases the fairness characteristic of the hashgraph algorithm as no node in the network is given any special permissions for deciding consensus time-stamp.

3.3 | Network intuition

Every mobile device be it be the service provider or consumer for offloading tasks is considered as a node in the hashgraph network. The mobile devices with low computation power need to offload their task or data to other mobile devices with high computation power. The gossip protocol is used to forward the need and state of each device in the network. Each node gossips the information that it has to the other nodes in the network that do not have that set of information. This way, the information is spread throughout the network in very little time. The transaction is differentiated with a flag that is set to "1" if the node wants to offload a task and "0" otherwise.[42] Hashgraph shares events or transactions to other nodes in the network through rounds, as shown in Figure 2. Table 2 shows the meaning of all the mathematical notations used in the next parts of this article.

Round number is incremented by one if the events generated in that round are gossiped to at least two-third of the total nodes in the network. The set of first event created by every node in every round is called as the witness for that round. To calculate the popularity, we see the witness of next round, that is, $k + 1$th round. In the hashgraph, the node n will calculate that which all witnesses in round $k + 1$ can see or connect to witness wit in round k. Witness wit + 1 is given a positive vote for the nodes whose witnesses in round k have a connection with wit + 1 and a negative vote for the

TABLE 2 Important notations and their meaning

Notation	Meaning
$\mathcal{M}$	Total mobile devices in network
$\mathcal{D}$	Data items present in mobile device
$\mathcal{S}$	Size of data items in $\mathcal{D}$
Γ	Computation time
$\mathcal{C}^L$	Local computation task
$\mathcal{T}$	Deadline time of data items in $\mathcal{D}$
Ω	Offloading opportunity
τ	Time required by Ω to offload
$\mathcal{Q}$	Total data that Ω can offload
ξ	Offloading solution
μ	Successful offloading probability
$\mathcal{W}(\xi)$	Welfare by offloading
$\epsilon(\xi)$	Offloading cost via cellular network
$\zeta(\xi)$	Offloading cost via WiFi network
w	Total cost for offloading
η	Offloading computation cost
t	Offloading computation time
δ	Win count by f mobile devices
Ψ	Objective value
$\mathcal{E}$	Mobile devices having equal counts
w_c	Weight associated with cost parameter
w_t	Weight associated with time parameter
v_c	Secret valuation for cost parameter
v_t	Secret valuation for time parameter
Δ_m^E	Weight of computation energy
Δ_m^T	Weight of computation time

nodes whose witnesses in round k do not have a connection with wit + 1. If two-third of the witnesses in round k give a positive vote for witness wit + 1, then wit + 1 is considered popular witness, else it is voted as a nonpopular witness. This way, although all the nodes are calculating the votes separately, still, they have the same consistent copy of DAG; all the votes are calculated as the same, and thus, it results into a 100% byzantine agreement. After a few rounds of voting, we can determine which witnesses are popular and which are not. The witness node at the round r is considered as a famous witness only if it could be seen by two-third of the witness nodes in the round $r + 1$. The event x in round $r + 1$ is said to be strongly seen by the event y in round $r - 1$ only if event x in round $r + 1$ has four ancestor events in round r and these all four ancestor events in round r will have the same ancestor as event y in the round $r - 1$. Hence, we can say that event x in the round $r + 1$ is strongly seen by the event y in round $r - 1$.

Now, we have to order all the events created in the network. The consensus of a transaction is calculated using the median of the time stamp at which the famous witnesses in that round received that particular transaction. This median is the consensus time stamp for this particular event. We do this for all the events in the graph, and the events are put in order for performing offloading tasks. Note that the median is the value that is nearest to the middle of all time stamps and is not changed or affected due to the extreme values. We take the median and not the mean because if some time stamps are too far from the middle value due to some reasons, and it will be discounted and will not affect the median value or the consensus time stamp for that event. Therefore, if a node tries to change its clock, it will not be able to change the consensus time stamp for the transactions.[42]

3.4 | Final ordering of the offloading tasks

The ascending order of events is generated based on hashgraph consensus time stamp. However, the time-stamp ordering might not always be the same as the ordering based on the deadline of the offloading task. The final order of the events is expected to include both hashgraph time stamp and deadline time of data items. Let us assume that there are n number of events present in the network. A particular weight can be set for prioritizing the order through a hashgraph consensus algorithm and the least deadline time. Let $\mathcal{Z}_1$ and $\mathcal{Z}_2$ be the vectors of events generated by mobile devices through hashgraph consensus algorithm and least deadline time, respectively. Let $\mathcal{W}_1$ and $\mathcal{W}_2$ be the weight associated with the hashgraph consensus order and least deadline time order, respectively. The values in vector $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are multiplied with the respective weights $\mathcal{W}_1$ and $\mathcal{W}_2$ and the results are stored in vectors $\mathcal{Y}_1$ and $\mathcal{Y}_2$, respectively,

$$\mathcal{Y}_1(i)_{i=1}^{n} = \mathcal{W}_1(\mathcal{Z}_1(i)_{i=1}^{n}), \tag{1}$$

$$\mathcal{Y}_2(j)_{j=1}^{n} = \mathcal{W}_2(\mathcal{Z}_2(j)_{j=1}^{n}). \tag{2}$$

The events corresponding to both vectors $\mathcal{Y}_1$, $\mathcal{Y}_2$ are the same. The values in vector $\mathcal{Y}_1$ are compared to values in vector $\mathcal{Y}_2$. If the first value in vector $\mathcal{Y}_1$ is smaller than all values in vector $\mathcal{Y}_2$, then that particular event in both the vectors $\mathcal{Y}_1$ and $\mathcal{Y}_2$ is eliminated, and the event is stored in another vector $\mathcal{X}$. Otherwise, if any value in vector $\mathcal{Y}_2$ is found smaller than the first value in vector $\mathcal{Y}_1$, then that particular event is eliminated, and the event is stored in another vector $\mathcal{X}$. The process is repeated until all the events in vector $\mathcal{Y}_1$ and $\mathcal{Y}_2$ are eliminated, and the vector $\mathcal{X}$ gives the final ordering of the offloading tasks considering both the hashgraph consensus time stamp and the least deadline time of data items.

4 | NETWORK MODEL FOR MOBILE DATA OFFLOADING

A network model is designed with a multiple number of mobile users that want to offload the computation task or can compute the tasks for other low computation devices. The set of mobile devices that create events in vector $\mathcal{X}$ is denoted by $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n\}$. Let m be the maximum number of data items present in each mobile device. The data items present in each mobile device be $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_m\}$. Let the size of these data items present in the mobile device be $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_m\}$. Any mobile device can have any data item that needs to be offloaded and that data item can have any size. Just for the ease of presentation, we have took the same variable for data item size and number of data items for all devices. The computation task can be either carried out locally by the mobile device itself or can be offloaded to another high computation device. This section explains the calculations related to the time and cost involved in both local computation and offloading.

4.1 | Local computation

The local computation cost refers to the cost of computation incurred to compute data item $\mathcal{D}_m$, $[m \in (1, m)]$ by mobile device $\mathcal{M}_n$, $[m \in (1, n)]$. Equation (4) shows the cost of computing the data item $\mathcal{D}_m$ in mobile device $\mathcal{M}_n$. Let the processing power of mobile device $\mathcal{M}_n$ be p_n and processing density of computing data item $\mathcal{D}_m$ be β_m.

Let Γ_m be the time required to compute a task of size $\mathcal{S}_m$ when done locally. The value of Γ_m can be calculated based on the following equation:

$$\Gamma_{\mathrm{m}} = \frac{\mathcal{S}_m \beta_m}{p_n}. \tag{3}$$

The cost of computing the task of size $\mathcal{S}_m$ locally, that is, $\mathcal{C}_m^L$ can be calculated as follows:[26]

$$\mathcal{C}_m^L = (\Delta_m^E \alpha_n + \Delta_m^T)\Gamma_m, \tag{4}$$

where α_n is a constant, and its value depends on the processor of mobile device $\mathcal{M}_n$. Δ_m^E and Δ_m^T denote the constant weights of computation energy and computation time of data item D_m, respectively. Their values lie in the range between 0 and 1 to avoid huge delays and depend upon the mobile device's computing capacity and type of task to compute.

4.2 | Offloading data items

The computation cost can be reduced by adopting a method of data offloading. Data offloading proposes to offload the computation task to some high computation device rather than performing it locally. Events are ordered in the hashgraph, and the data items in the mobile devices are offloaded to other mobile devices as pictorially shown in Figure 3. The deadline constraint of the offloading task is also considered.

Let data items $\mathcal{D} = \{D_1, D_2, ..., D_m\}$ present in mobile devices $\mathcal{M}_n$ be associated with a deadline time denoted by $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_m\}$. All the data items in mobile devices are required to be computed before the deadline time. It is assumed that all the mobile devices in the network are capable of transferring the data through the wireless network with the help of Bluetooth, Wi-Fi, and so on. Mobile devices can also directly offload the data item through the cellular network. Let C be the cost of offloading the data directly through the cellular network. If all the mobile devices that have to offload the data item opt to offload through the cellular network, it will end up into congestion in the cellular network. To reduce the congestion in cellular network, mobile devices offload the data item through a Wi-Fi network. Let c be the cost of offloading the task through a Wi-Fi network. In a Wi-Fi network, the task is offloaded through offloading opportunities. Offloading opportunity refers to the ability of the carrier to perform the offloading task.

The offloading opportunity is denoted by $\Omega = \langle \tau, \mathcal{P}, \mathcal{Q} \rangle$, where τ represents the time in which the data can be offloaded, $\mathcal{P}$ represents the probability of Wi-Fi network providing offloading service, and $\mathcal{Q}$ represents the offloading capacity of that particular offloading opportunity. Let s be the total number of offloading opportunities provided by the Wi-Fi network. There are a total of $\{\Omega_1, \Omega_2, ..., \Omega_s\}$ offloading opportunities. All the m data items in the mobile device are offloaded through offloading opportunities provided by the Wi-Fi network to reduce the congestion in the cellular network.

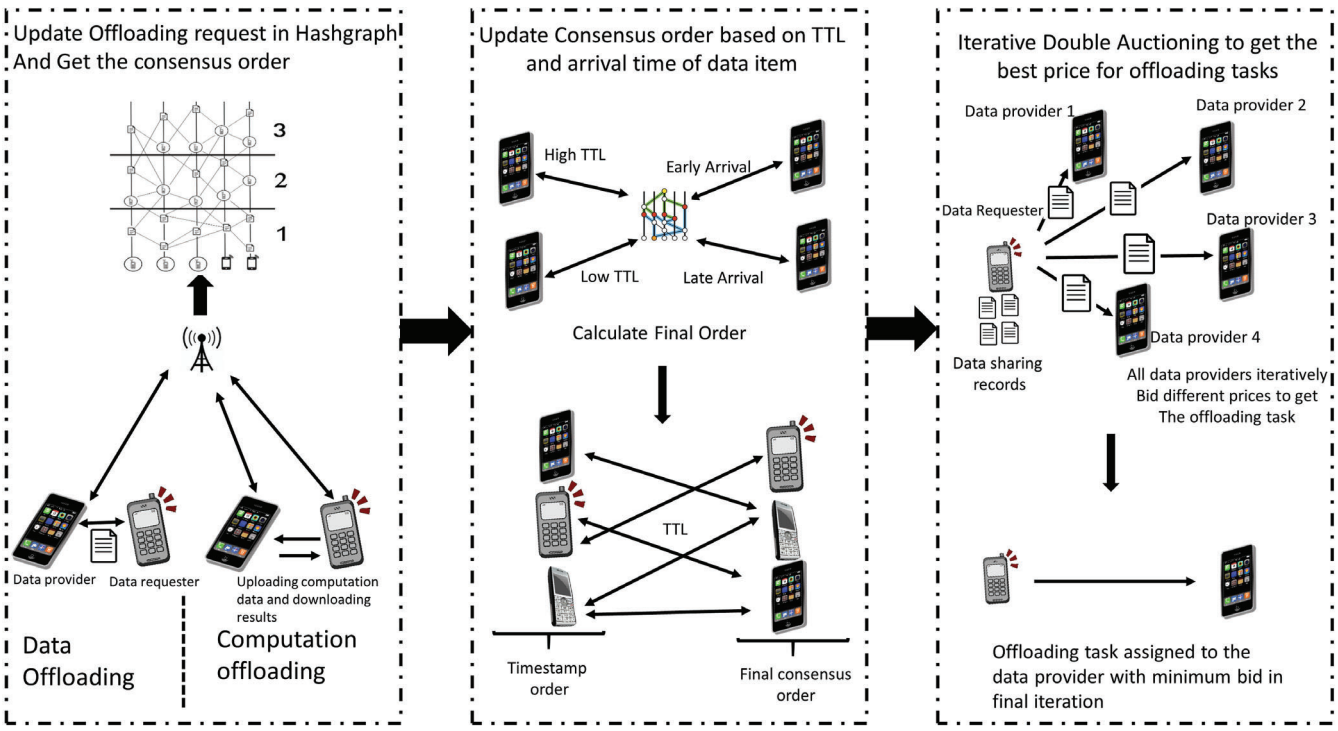


FIGURE 3 A step-wise description of proposed model including ordering of events and offloading the tasks

Algorithm 1. Task Offloading Algorithm

Input: Offloading Solution ξ

$\xi = \{\xi_1, \xi_2, ..., \xi_v\}$

Output: Cost w for offloading m tasks

1: **for** $v = 1 : v$ **do**
2: **for** $m = 1 : m$ **do**
3: **for** $s = 1 : s$ **do**
4: **if** $\mathcal{T}_m \geq \tau_s$ **then**
5: $\xi_v = 1$
6: **end if**
7: **if** $\xi_v = 1$ **then**
8: $\mu_m(\xi) = 1 - \prod_{s:(\mathcal{D}_m, \Omega_s) \in \xi} (1 - \mathcal{P}_s)$
9: $\mathcal{W}_v(\xi) = \sum_{m=1}^{m} \mathcal{S}_m * \mu_m(\xi)$
10: **if** $\mathcal{S}_m \leq \mathcal{Q}_s$ **then**
11: $\epsilon_v(\xi) = C \sum_{m=1}^{m} \mathcal{S}_m(1 - \mu_m(\xi))$
12: $\zeta_v(\xi) = c \sum_{m=1}^{m} \mathcal{S}_m * \mu_m(\xi)$
13: $w = \epsilon_v(\xi) - \zeta_v(\xi)$
14: $\mathcal{Q}_s = \mathcal{Q}_s - \mathcal{S}_m$
15: $\xi_v = 0$
16: **end if**
17: **else if** $\xi_v < \xi_{v-1}$ **then**
18: $\xi_{v-1} = 1$
19: $\mathcal{Q}_s = \mathcal{Q}_s + \mathcal{S}_{m-1}$
20: **if** $\mathcal{S}_s \leq \mathcal{Q}_s$ **then**
21: $\mathcal{Q}_s = \mathcal{Q}_s - \mathcal{S}_m$
22: $\xi_v = 0$
23: **end if**
24: **end if**
25: **end for**
26: **end for**
27: **end for**

The capacity in all the offloading opportunities is to be utilized optimally by offloading data items according to their deadline constraint. Data items are arranged in the order calculated in the previous section, which is a combination of arrival time stamp and deadline time. Any offloading opportunity can offload data item only if its offloading time is less than or equal to the deadline time ($\tau \leq \mathcal{T}$) and capacity of offloading opportunity is greater than or equal to the size of data item ($\mathcal{Q} \geqslant \mathcal{S}$). The particular offloading opportunity Ω_s, $[s \in (1, s)]$, and its associated data item $\mathcal{D}_m$ are stored in vector named offloading solution ξ. Let v be the total number of offloading solutions. ξ_v will be -1 initially, and its value will change to 1 if offloading opportunity satisfies the deadline constraint of data item, that is, ($\tau \leq \mathcal{T}$) and its value will change to 0 if both deadline constraint and data size constraint, that is, ($\mathcal{Q} \geqslant \mathcal{S}$) are fulfilled. Algorithm 1 shows the detailed steps involved in the calculation of computation cost. The calculated cost is used as an input in Algorithm 2. Let there be v offloading solutions present to offload data items. For each data item of different sizes, we compare the offloading time given by the opportunity with the deadline time associated with that data item as shown in line number 4 of Algorithm 2. If the deadline time is less than or equal to the time given by offloading opportunity, then the data item is assigned to the respective opportunity. This is done by setting the flag of offloading opportunity to 1 as shown in line 5 of Algorithm 2. If the flag is 1, then we calculate the successful offloading probability and welfare, as shown in lines 8 and 9 of Algorithm 2. Furthermore, we verify the size of data item and compare it with the size given by offloading opportunity. If the size of data item is less than the size provided by offloading opportunity, the offloading cost is calculated as shown in line 13 of Algorithm 2. The flag is again set back to zero ones the data is offloaded.

The successful offloading probability that tells about the probability of offloading a data item through offloading opportunity is denoted by μ_m. This probability is further used to calculate the offloading cost

$$\mu_m(\xi) = 1 - \prod_{s:(\mathcal{D}_m,\Omega_s)\in\xi} (1 - \mathcal{P}_s). \tag{5}$$

Now, the welfare of offloading data item through offloading opportunity is calculated. The welfare function denoted by $\mathcal{W}(\xi)$ is expressed in the following equation:

$$\mathcal{W}_v(\xi) = \sum_{m=1}^{m} \mathcal{S}_m * \mu_m(\xi). \tag{6}$$

Data item is offloaded through offloading opportunity in a way that the overall welfare value is maximum. The cost of offloading data item by offloading opportunity having the maximum welfare value is denoted by w. This value includes the cost of offloading through cellular network $\varepsilon_v(\xi)$ and Wi-Fi network $\zeta_v(\xi)$

$$w = \varepsilon_v(\xi) - \zeta_v(\xi). \tag{7}$$

The offloading cost through cellular network and Wi-Fi network can be calculated in the following two equations:

$$\varepsilon_v(\xi) = C \sum_{m=1}^{m} \mathcal{S}_m(1 - \mu_m(\xi)), \tag{8}$$

$$\zeta_v(\xi) = c \sum_{m=1}^{m} \mathcal{S}_m * \mu_m(\xi), \tag{9}$$

where C refers to the cost of offloading through cellular network per data item and c refers to the cost of offloading through Wi-Fi. The amount of data being offloaded through Wi-Fi or cellular network depend on the successful offloading probability as calculated in Equation (5). Based on this, we derive Equations (8) and (9). Equation (8) uses the probability of data not getting offloaded through Wi-Fi and Equation (9) uses the probability of data getting offloaded through Wi-Fi. Hence, the offloading cost calculated is the optimal solution that keeps the consensus time-stamp and deadline constraint into consideration simultaneously.

5 | GAME THEORY FOR BARGAINING COST AND TIME

Low computation devices offload their data items to the high computation devices, and the cost for this offloading is calculated according to Algorithm 1. Mobile devices with high computation power do not know the cost given by other high computation devices. A game theory model is used to bargain the computation cost for low computation devices. Let $\mathcal{M}_n$ be any mobile device with low computation power, which wants to offload its task to any other high computation mobile devices in $\mathcal{M}$. Let f be the number of high computation devices that can offload the task of $\mathcal{M}_n$. The costs for offloading the tasks given by these f mobile devices are stored in vector $\eta = \{\eta_1, \eta_2, ..., \eta_f\}$, and the time for computing the tasks given by these f mobile devices is stored in vector $t = \{t_1, t_2, ..., t_f\}$. The cost given by $\mathcal{M}_n$ for computing its m tasks is given by w, as shown in Equation (7). There is a possibility that the cost w given by low computation devices is too less than the cost expected or demanded by high computation devices for offloading η. The value of w must be near to the median of the values present in vector η to maintain the fair auction process. To do so, we first calculate the median index in the cost vector η as follows. The same computation is done in line 2 of Algorithm 2.

$$\text{mp} = \frac{f+1}{2}. \tag{10}$$

Algorithm 2. Algorithm for Game Theory Model

Input: Cost w calculated through algorithm 1
Output: Win count δ of mobile devices with high computation power
$\delta = \{\delta_1, \delta_2, ..., \delta_f\}$

1: w is the cost given by mobile device $\mathcal{M}_n$
2: $mp = \frac{f+1}{2}$
3: **while** $(\eta_{mp} - w) = negligible$ **do**
4: $w = w + \chi$
5: **end while**
6: Initialize $b = 0$
7: **while** $|\Psi^f_{f=1,b} - \Psi^f_{f=1,b-1}| \leq \varkappa$ **do**
8: $b = b + 1$
9: **for** $f = 1 : f$ **do**
10: $\eta^f_{f=1} = \frac{\eta_f - min(\eta)}{max(\eta) - min(\eta)}(\mathcal{Y} - \mathcal{X}) + \mathcal{X}$
11: $t^f_{f=1} = \frac{t_f - min(t)}{max(t) - min(t)}(\mathcal{V} - \mathcal{U}) + \mathcal{U}$
12: $\Psi^f_{f=1} = (w_c * \eta_f) + (w_t * t_f)$
13: **end for**
14: **if** $\Psi^f_{f=1} = min(\Psi)$ **then**
15: $\delta_f = \delta_f + 1$
16: **end if**
17: **for** $f = 1 : f$ **do**
18: **if** $\eta^f_{f=1} \neq min(\eta)$ and $\eta^f_{f=1} \geqslant v_c$ **then**
19: $\eta^f_{f=1} = \eta_f - \theta$
20: **end if**
21: **if** $t^f_{f=1} \neq min(t)$ and $t^f_{f=1} \geqslant v_t$ **then**
22: $t^f_{f=1} = t_f - \kappa$
23: **end if**
24: **end for**
25: **end while**

Now, we iteratively increase the value of w by a constant χ to bring it near to value of η_{mp} as shown in lines 3 and 4 in Algorithm 2. The purpose is to bring the difference between η_{mp} and w near to negligible.

$$w = w + \chi. \tag{11}$$

5.1 | Offloading cost optimization

In this section, we discuss the bargaining of cost between f mobile devices. To compare the bids of f devices, a single objective value Ψ that takes into consideration both the time and cost parameters is formulated. It is worth noting that the low computation device users can assign different weights to cost and time parameters. To calculate the value of Ψ, we need to calculate the normalized time and cost values for all f devices as follows. These steps are also shown in lines 10 to 12 in Algorithm 2.

$$\eta^f_{f=1} = \frac{\eta_f - \min(\eta)}{\max(\eta) - \min(\eta)}(\mathcal{Y} - \mathcal{X}) + \mathcal{X}, \tag{12}$$

$$t^f_{f=1} = \frac{t_f - \min(t)}{\max(t) - \min(t)}(\mathcal{V} - \mathcal{U}) + \mathcal{U}, \tag{13}$$

where $\mathcal{X}$ and $\mathcal{Y}$ are the allowed minimum and maximum values of cost and $\mathcal{U}$ and $\mathcal{V}$ are the allowed minimum and maximum values of time, respectively.

Using the above two equations, we can calculate the final objective value Ψ based on the following equation:

$$\Psi^{f}_{f=1} = (w_c * \eta_f) + (w_t * t_f), \tag{14}$$

where w_c and w_t denote the weights given to cost and time parameter by low computation device users. The objective values calculated for each high computation device are stored in vector $\Psi = \{\Psi_1, \Psi_2, ..., \Psi_f\}$.

The high computation mobile device with a minimum value of Ψ is declared as the winner for the offloading task as shown in line 14 of Algorithm 2. This information is shared on the network. The win count of f devices is stored in vector δ to calculate the final winner. The other mobile devices tend to reduce their cost and time values in order to win the offloading task. The process continues over b number of iterations. The value of b is determined based on the stopping condition, as discussed below. The stopping condition is also shown in line 7 of Algorithm 2.

$$(\Psi^{f}_{f=1,b} - \Psi^{f}_{f=1,b-1}) \leq \varkappa, \tag{15}$$

where $\varkappa$ is very small constant. However, the Ψ vector can have multiple minimum values. In such a case, all these devices are considered as winners and do not update their cost and time values in the next iteration.

The values in η and t vectors are decreased by very small values θ and κ, respectively, only for those devices that are not the winners and when their cost and time values are not less than their secret valuation. These updating conditions are depicted in Equations (16) and (17), respectively,

$$\eta^{f}_{f=1} \neq \min(\eta), \eta^{f}_{f=1} \geqslant v_c, \tag{16}$$

$$t^{f}_{e=1} \neq \min(t), t^{f}_{e=1} \geqslant v_t, \tag{17}$$

where v_c and v_t are the valuation for cost and time, respectively. The secret valuation is the actual budget of the high computation devices below which it is not possible for them to perform offloading.

Algorithm 3. Algorithm for Final Allotment

Input: Win count δ of mobile devices with high computation power
$\delta = \{\delta_1, \delta_2, ..., \delta_f\}$
Output: Mobile device with low computation power is allotted to mobile device with high computation power
$\mathcal{M}_n \iff \mathcal{M}_\Lambda$

1: **if** Only one mobile device with high computation power has maximum win count: **then**
2: **for** $f = 1 : f$ **do**
3: **if** $\delta^{f}_{f=1} = max(\delta)$ **then**
4: $\Lambda = f$
5: $\mathcal{M}_n \iff \mathcal{M}_\Lambda$
6: **end if**
7: **end for**
8: **else**
9: **for** $f = 1 : f$ **do**
10: **if** $\mathcal{E}^{f}_{f=1} = min(\mathcal{E})$ **then**
11: $\Lambda = f$
12: $\mathcal{M}_n \iff \mathcal{M}_\Lambda$
13: **end if**
14: **end for**
15: **end if**

5.2 | Allocation of mobile device

In this section, we finally allocate the mobile device with low computation power $\mathcal{M}_n$ to the best possible mobile device with high computation power. When the stopping condition as shown in Equation (15) is satisfied, mobile device that has won the maximum number of iterations is allocated to $\mathcal{M}_n$ mobile device to offload its tasks according to Equation (18)

$$\mathcal{M}_n \Leftrightarrow \mathcal{M}_\Lambda, \tag{18}$$

where $\Lambda = f \in [\delta^f_{f=1} = \max(\delta)]$ and δ is the number of iterations won by a particular device.

However, if two or more mobile devices have an equal maximum win count then the mobile device with least cost among them will be allocated, as shown in lines 9 to 12 in Algorithm 3. Let $\mathcal{E} = \{\mathcal{E}_1, ..., \mathcal{E}_f\}$ be the vector having cost of mobile devices that have maximum equal win counts. Then, the mobile device with least cost among mobile devices in $\mathcal{E}$ vector will be allocated to mobile device $\mathcal{M}_n$.

$$\mathcal{M}_n \Leftrightarrow \mathcal{M}_\Lambda, \tag{19}$$

where $\Lambda = f \in [\mathcal{E}^f_{f=1} = \min(\mathcal{E})]$.

6 | NUMERICAL ANALYSIS

In this section, simulations are conducted to show the efficiency and performance of our proposed algorithm for the mobile data offloading problem.

6.1 | Simulation settings

We consider that a mobile device with low computation power wants to compute some tasks. For the purpose of evaluation and simulation, we assume the size of the task in a mobile device with low computation power to be in the range of [30 000, 40 000] bytes. If the task is computed locally, it takes more time and cost for its computation compared with offloading the task to other mobile devices with high computation power. The constant weight for computation energy and computation time in mobile devices are assumed to be within intervals of [0, 1]. The value of processing density of low computation devices is taken in the range of [200, 600]. The deadline time associated with tasks is assumed to be in the range of [15, 25] seconds. Similarly, the offloading time associated with offloading opportunities is assumed to be within intervals of [12, 22] seconds. The cost of offloading through cellular network and Wi-Fi network is taken as 4 and 1 dollar, respectively. Ten high computation devices are considered to be participating in the game theory. The values of cost and time given by high computation devices for task computation lie in the range of [17, 42] cents and [7, 17] seconds, respectively. The weight associated with cost and time parameters of high computation devices is taken as 0.7 and 0.3, respectively.

The entire code for hashgraph simulation and interaction between different nodes in the distributed network has been written in python. The main focus of this work is to present the effectiveness of the use of distributed ledger in mobile offloading. There is no centralized authority involved to perform the offloading task. A smart contract is used to perform the negotiation between high and low computation devices. Initially, the code was tested on a CPU, and all the computation cost and time were calculated. Finally, the NVIDIA Titan Xp GPU was used to verify the results in the case of a large number of users. MATLAB tool was also used to create the graphs in a more presentable manner.

6.2 | Performance evaluation

Figure 4 shows the increase in computation cost for computing tasks locally by different mobile devices with low computation power. We can observe from the graph that MD1 (Mobile Device 1) took less computation cost than MD2 (Mobile Device 2) and MD3 (Mobile Device 3) for the same size of task because MD1 has a better processor as compared to MD2 and MD3. As the size of the task increases, its computation cost also increases for every mobile device. Similarly, Figure 5

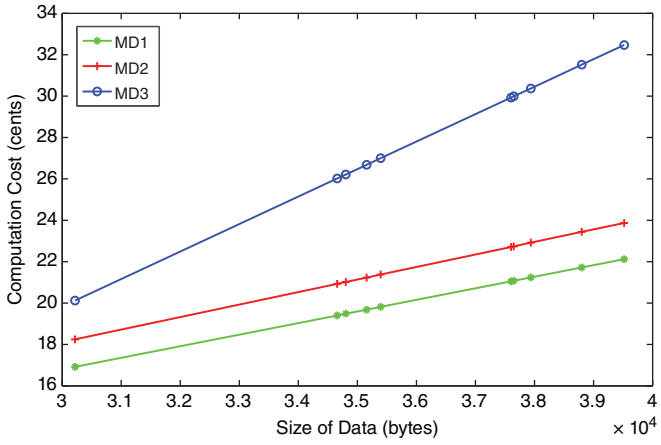


FIGURE 4 Computation cost vs size of data for three different mobile devices

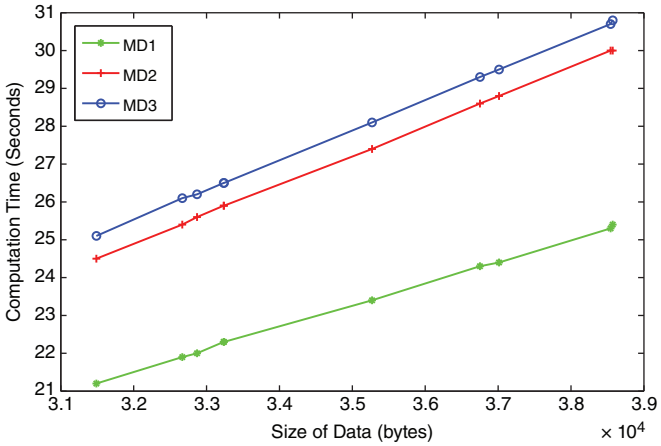


FIGURE 5 Computation time vs size of data for three different mobile devices

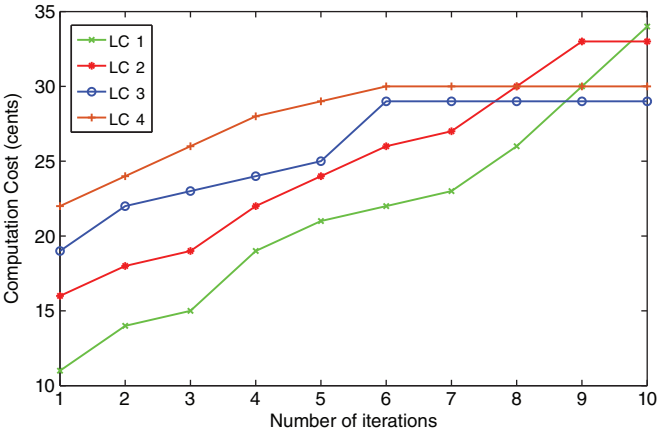


FIGURE 6 Cost given by different low computation devices

shows the increase in computation time for computing tasks locally by different mobile devices with low computation power. It can be observed that MD1 took less computation time than MD2 and MD3 for the same size of the task due to its better processing power. As the size of the task increases, its computation time also increases for every mobile device.

Figure 6 shows the variation of the price offered by low computation devices for task computation. It can be observed that the prices keep on increasing until the price value comes near the median of costs given by mobile devices with high computation power over iterations. Once the price comes in the range of price given by high computation devices, the growth in the prices stops.

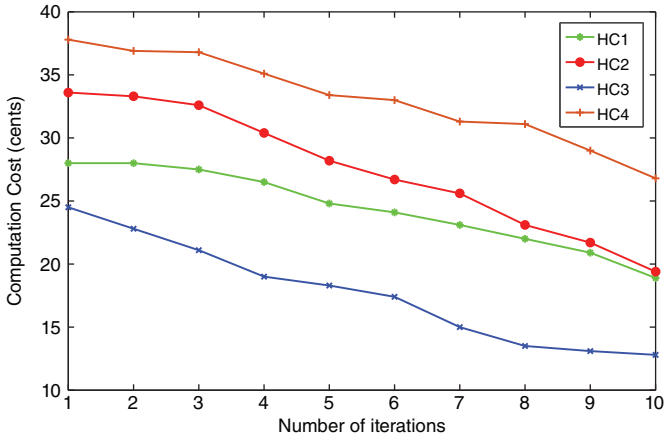


FIGURE 7 Variation of cost by high computation devices over iterations

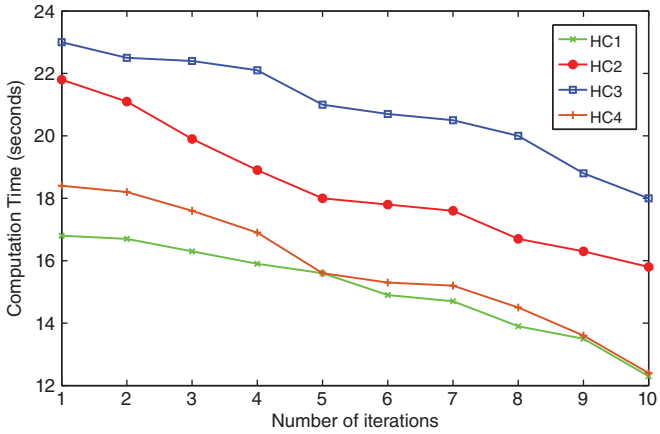


FIGURE 8 Variation of time by high computation devices over iterations

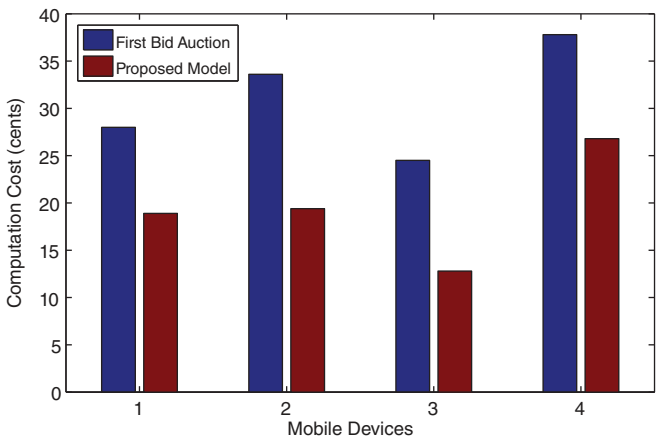


FIGURE 9 Comparison of computation cost between the first bid and proposed approaches

Figure 7 shows the variation in the prices offered by high computation devices over iterations to win the computation tasks. The high computation devices decrease the price until they reach their threshold values. For a particular iteration, a mobile device with high computation power updates its computation cost only if it is not a winner in the previous iteration to win the present iteration. Figure 8 shows how the mobile devices with high computation power update their computation time values iteration wise until they reach their threshold values. For a particular iteration, a mobile device with high computation power updates its computation time only if it is not a winner in the previous iteration to win the present iteration.

Figure 9 shows the comparison of computation cost given by four different mobile devices with high computation power in the proposed model and the first bid auction model. It can be observed that the low computation devices are

FIGURE 10 Comparison of computation time between the first bid and proposed approaches

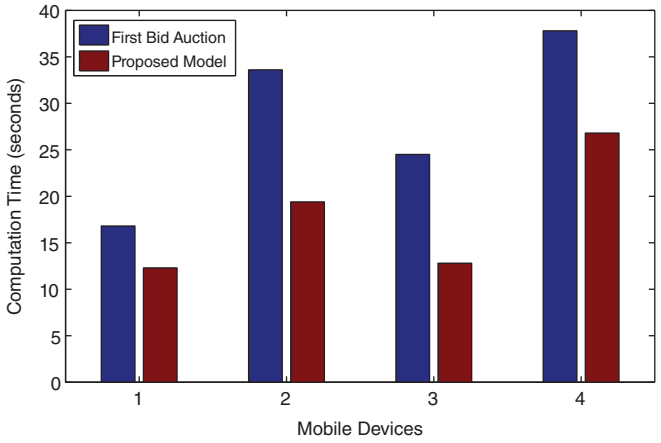


FIGURE 11 Winning counts of different mobile devices over iterations

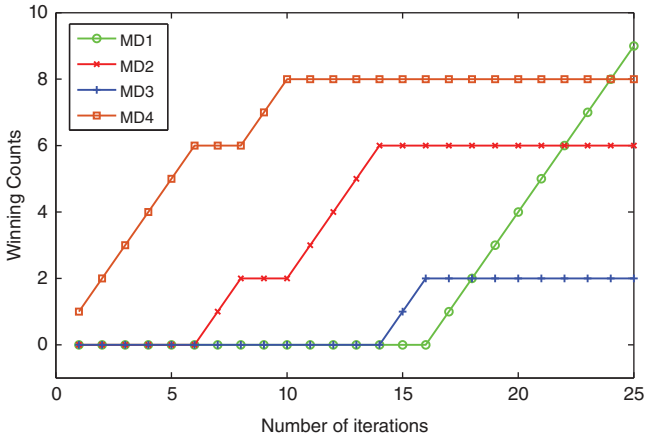
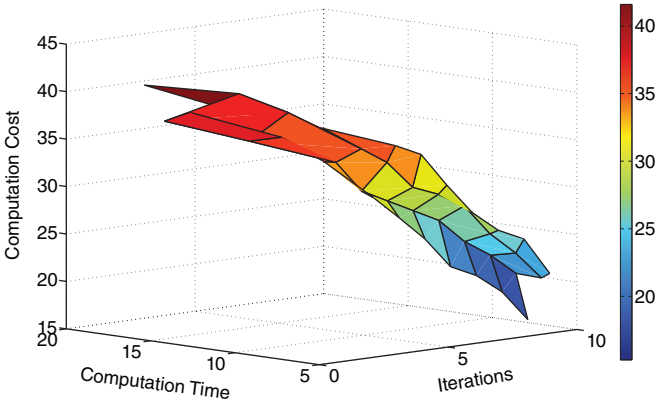


FIGURE 12 Variations of computation cost and time over iterations of different mobile devices



able to get the computation task done at a lower price in the proposed model compared with the other models. Similarly, Figure 10 shows the comparison of computation time given by four different mobile devices with high computation power in the proposed model and the first bid auction model.

Figure 11 shows the win counts of mobile devices with high computation power over iterations. For a particular iteration, a mobile device with high computation power that has less computation cost and time compared with other mobile devices is declared as a winner, and its win count is increased by the value 1. It can be observed that in beginning iterations, MD4 (Mobile Device 4) is winning the bid, but finally, MD1 comes out to be the winner for the computation task as it has the minimum valuation.

Figure 12 shows the variation in computation cost and time of high computation mobile devices over iterations. It can be observed that the computation cost and time of mobile devices with high computation power reduce and reach

their threshold in order to win the bid. Finally, at the end of the process, the mobile device with high computation power, which has a maximum win count value or minimum computation cost and time, is declared as the final winner and is allowed to offload. The color map on the right shows the values of computation cost and relative changes in the colors. As the iterations are increasing, the computation cost and time are decreasing. Therefore, the proposed model is proven to enhance the utility and welfare of mobile devices.

7 | CONCLUSION

In this article, we explained the drawbacks of blockchain and why we selected hashgraph for establishing a distributed P2P network. The low-computation devices offload their tasks to high computation devices. The offloading of tasks is scheduled, considering both the hashgraph consensus time stamp and the least deadline time of tasks. We proposed a game theory, where high computation devices participate in the auction mechanism that results in the computation of task in cost-optimal and time-efficient manner compared with its counterparts.

ACKNOWLEDGEMENTS

The authors acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. They also thank D. N. G. Krishna for his support in the completion of the project as an intern at BITS, Pilani.

ORCID

Vinay Chamola https://orcid.org/0000-0002-6730-3060

REFERENCES

1. Cisco. Cisco Visual Networking Index: Global mobile data traffic forecast update. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html. Accessed February 18, 2019.
2. Li Y, Shi G, Yin W, Liu L, Han Z. A distributed ADMM approach with decomposition-coordination for mobile data offloading. *IEEE Trans Veh Technol*. 2017;67(3):2514-2530.
3. Baccarelli E, Cordeschi N, Mei A, Panella M, Shojafar M, Stefa J. Energy-efficient dynamic traffic offloading and reconfiguration of networked data centers for big data stream mobile computing: review, challenges, and a case study. *IEEE Netw*. 2016;30(2):54-61.
4. Yu H, Cheung MH, Iosifidis G, Gao L, Tassiulas L, Huang J. Mobile data offloading for green wireless networks. *IEEE Wirel Commun*. 2017 Aug;24(4):31-37.
5. Aijaz A, Aghvami H, Amani M. A survey on mobile data offloading: technical and business perspectives. *IEEE Wirel Commun*. 2013;20(2):104-112.
6. Zhou H, Wang H, Li X, Leung VC. A survey on mobile data offloading technologies. *IEEE Access*. 2018;6:5101-5111.
7. Liu D, Khoukhi L, Hafid A. Prediction-based mobile data offloading in mobile cloud computing. *IEEE Trans Wirel Commun*. 2018;17(7):4660-4673.
8. Yu S, Langar R, Fu X, Wang L, Han Z. Computation offloading with data caching enhancement for mobile edge computing. *IEEE Trans Veh Technol*. 2018;67(11):11098-11112.
9. Neto JLD, Yu SY, Macedo DF, et al. ULOOF: a user level online offloading framework for mobile edge computing. *IEEE Trans Mob Comput*. 2018;17(11):2660-2674.
10. Shah-Mansouri H, Wong VW, Huang J. An incentive framework for mobile data offloading market under price competition. *IEEE Trans Mob Comput*. 2017;16(11):2983-2999.
11. Wang K, Lau FC, Chen L, Schober R. Pricing mobile data offloading: a distributed market framework. *IEEE Trans Wirel Commun*. 2015;15(2):913-927.
12. Tang W, Zhao X, Rafique W, Dou WA. Blockchain-based offloading approach in fog computing environment. Paper presented at: 2018 IEEE International Conference on Parallel and Distributed Processing with Applications, Ubiquitous Computing and Communications, Big Data and Cloud Computing, Social Computing and Networking, Sustainable Computing and Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom); IEEE; 2018:308-315.
13. Zhao N, Wu H, Chen Y. Coalition game-based computation resource allocation for wireless blockchain networks. *IEEE Internet Things J*. 2019;6:5.
14. Hassija V, Chamola V, Saxena V, Jain D, Goyal P, Sikdar B. A survey on IoT security: application areas, security threats, and solution architectures. *IEEE Access*. 2019;7:82721-82743.
15. Hassija V, Bansal G, Chamola V, Saxena V, Sikdar B. BlockCom: a blockchain based commerce model for smart communities using auction mechanism. Paper presented at: 2019 IEEE International Conference on Communications Workshops (ICC Workshops); 2019:1-6.
16. Hassija V, Saxena V, Chamola V. Scheduling drone charging for multi-drone network based on consensus time-stamp and game theory. *Comput Commun*. 2019;149:51-61.
17. Rimal BP, Maier M. Mobile data offloading in FiWi enhanced LTE-A heterogeneous networks. *J Opt Commun Netw*. 2017;9(7):601-615.

18. Gao G, Xiao M, Wu J, Han K, Huang L, Zhao Z. Opportunistic mobile data offloading with deadline constraints. *IEEE Trans Parall Distr Sys*. 2017;28(12):3584-3599.
19. Mehmeti F, Spyropoulos T. Performance modeling, analysis, and optimization of delayed mobile data offloading for mobile users. *IEEE/ACM Trans Netw*. 2017;25(1):550-564.
20. Flores H, Hui P, Nurmi P, et al. Evidence-aware mobile computational offloading. *IEEE Trans Mob Comput*. 2017;17(8):1834-1850.
21. Cheon HR, Kim JH. Social context-aware mobile data offloading algorithm via small cell backhaul networks. *IEEE Access*. 2019;7:39030-39040.
22. Li M, Quek TQ, Courcoubetis C. Mobile data offloading with uniform pricing and overlaps. *IEEE Trans Mob Comput*. 2019;18(2):348-361.
23. Zhang X, Guo L, Li M, Fang Y. Motivating human-enabled mobile participation for data offloading. *IEEE Trans Mob Comput*. 2017;17(7):1624-1637.
24. Hu Z, Lu Z, Wen X, Li Q. Stochastic-geometry-based performance analysis of delayed mobile data offloading with mobility prediction in dense IEEE 802.11 networks. *IEEE Access*. 2017;5:23060-23068.
25. Xu D, Li Q, Zhu H. Energy-saving computation offloading by joint data compression and resource allocation for mobile-edge computing. *IEEE Commun Lett*. 2019;23(4):704-707.
26. Chamola V, Hassija V, Sikdar B, Kumar N, Ansari N. Energy and latency aware resource management for solar powered cellular networks. *IEEE Netw*. 2019.
27. Miglani A, Kumar N, Chamola V, Zeadally S. Blockchain for Internet of energy management: a review, solutions and challenges. *Comp Commun*. 2019.
28. Hassija V, Chamola V, Garg S, Kaddoum G, Jayakody NA. Blockchain-based framework for lightweight data sharing and energy trading in V2G network. *IEEE Trans Veh Technol*. 2019.
29. Bansal G, Hassija V, Chamola V, Kumar N, Guizani M. Smart stock exchange market: a secure predictive decentralised model. Paper presented at: IEEE Globecom; Waikoloa, HI; December 2019:1-6.
30. Cho H. ASIC-resistance of multi-hash proof-of-work mechanisms for blockchain consensus protocols. *IEEE Access*. 2018;6:66210-66222.
31. Rosa R, Rothenberg CE. Blockchain-based decentralized applications for multiple administrative domain networking. *IEEE Commun Stand Mag*. 2018;2(3):29-37.
32. Dunphy P, Petitcolas FAP. A first look at identity management schemes on the blockchain. *IEEE Secur Priv*. 2018;16(4):20-29.
33. Chamola V, Tham CK, Chalapathi GS. Latency aware mobile task assignment and load balancing for edge cloudlets. Paper presented at: 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops); IEEE; 2017:587-592.
34. Chalapathi GSS, Chamola V, Tham CK, Gurunarayanan S, Ansari N. An optimal delay aware task assignment scheme for wireless SDN networked edge cloudlets. *Futur Gener Comput Syst*. 2020;102:862-875.
35. Mohanta BK, Panda SS, Jena D. An overview of smart contract and use cases in blockchain technology. Paper presented at: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT); 2018:1-4.
36. Zou J, Ye B, Qu L, Wang Y, Orgun MA, Li LA. Proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services. *IEEE Trans Serv Comput*. 2018;12(3):429-445.
37. O'Neill M, Robshaw MJB. Low-cost digital signature architecture suitable for radio frequency identification tags. *IET Comput Digit Tech*. 2010 January;4(1):14-26.
38. Yang Z, Lang W, Tan Y. Fair micropayment system based on hash chains. *Tsinghua Sci Technol*. 2005;10(3):328-333.
39. Alladi T, Chamola V, Parizi R, Choo KKR. Blockchain applications for industry 4.0 and industrial IoT: a review. *IEEE Access*. 2019;7:176935-176951.
40. Alladi T, Chamola V, Rodrigues JJ, Kozlov SA. Blockchain in smart grids: a review on different use cases. *Sensors*. 2019;19(22):4862.
41. Kus Khalilov MC, Levi AA. Survey on anonymity and privacy in bitcoin-like digital cash systems. *IEEE Commun Surv Tut*. 2018;20(3):2543-2585.
42. Leemon Baird MH, Madsen P. Hedera: A Public Hashgraph Network and Governing Council. https://www.hedera.com/whitepaper. Accessed February 9, 2019.
43. Wang S, Ouyang L, Yuan Y, Ni X, Han X, Wang F. Blockchain-enabled smart contracts: architecture, applications, and future trends. *IEEE Trans Syst Man Cybern Syst*. 2019;49(11):266-2277.

How to cite this article: Hassija V, Saxena V, Chamola V. A mobile data offloading framework based on a combination of blockchain and virtual voting. *Softw Pract Exper*. 2021;51:2428–2445. https://doi.org/10.1002/spe.2786