2023 IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom) | 979-8-3503-3333-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/MetaCom57706.2023.00078

ATOM : A Decentralized Task Offloading Framework for Mobile Edge Computing through Blockchain and Smart Contracts

Roshan Singh, Debanjan Roy Chowdhury, Sukumar Nandi and Sunit Kumar Nandi
Department of CSE, Indian Institute of Technology, Guwahati, India
roshancsofficial@gmail.com, chowdhur@iitg.ac.in, sukumar@iitg.ac.in,sunitnandi834@gmail.com

***Abstract*—Edge computing is the paradigm that offers low latency services by bringing computation and data storage closer to data sources. Due to the resource-constrained nature of the edge devices, they may not be able to handle all the computation tasks independently. In such cases, task offloading to the peer edge devices is looked for. However, as the devices are from different manufacturers, collaboration among them is challenging due to the lack of any trusted common platform. Blockchain along with smart contracts can help bridge the gap by providing a common decentralized platform of mutual trust. In this work, we introduce a permissioned blockchain-based platform for decentralized computation task offloading among edge devices. A smart contract is designed which provides an ecosystem of mutual trust by ensuring rewards for the honest participants and punishment for the misbehaving participants. Our proposed scheme also does fractional task offloading to encourage the participation of resource-constrained devices. We evaluated our scheme through a proof of concept implementation over an Ethereum-based testbed and found that our scheme can provide a trust-worthy eco-system for resource-constrained edge devices.**

Keywords: Mobile Edge Computing, Permissioned Blockchain, Reputation Management in MEC

I. Introduction

Mobile Edge Computing (MEC) [1] enables computing services to be extended to the edge of the network through base stations and access points [2]. Edge devices do suffer from constrained computing capabilities in comparison to the cloud and may not be able to perform all the computations on their own when they operate individually. Under such scenarios, user tasks can be divided into subtasks and offloaded to peer-edge devices leveraging the available computational resources of the peer-edge devices. However, in a MEC environment edge devices might be from different manufacturers having varying computational as well as storage capabilities. An edge device of one manufacturer may deny processing computation requests coming from edge devices of other manufacturers. Incentivizing/rewarding edge devices for processing computation requests may motivate them to participate in the task offloading eco-system. Another strategic challenge while assigning a task to an end device is to decide the most appropriate parameters for task offloading. A device having minimum latency might not be always the best to process the task as in the MEC setup a misbehaving device might be much closer than other devices. Such situations ask for a proper and effective decentralized reputation management mechanism so that misbehaving edge devices should not get an offloaded task. However, realizing such a trusted decentralized ecosystem is a challenging task due to the reasons of - *Node heterogeneity*, *Task heterogeneity*, *Dynamic node availability*, *Lack of trust*.

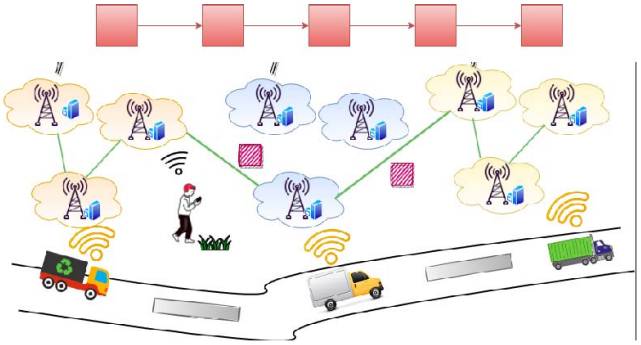

Fig. 1. Proposed Blockchain-Based Mobile Edge Computing Framework

Motivated by the facts, we propose the solution "***A** Decentralized **T**ask **O**ffloading Framework for **M**obile Edge Computing through Blockchain and Smart Contracts*" (ATOM). The contributions of the proposed work are highlighted as follows.

1) This work proposes a common task offloading platform for MEC where trust is enforced through smart contracts on the blockchain. In the proposed system, honest behavior is the default strategic choice of the participants as they incur an economic loss for any kind of misbehavior.
2) This is the first time, the computation results submitted by computing nodes are validated before using it by the task offloading node to avoid adverse effects. The reliability of the results is achieved through the proposed stake-based task bidding scheme and the reputation score management system through blockchain.
3) Unlike other works, this work eliminated the denial of reward payment to honest computing nodes, and misbehaving nodes are economically penalized. Also, a fractional task offloading strategy is introduced which reduces the starvation time for the availability of computing edge nodes with sufficient large resource availability.

979-8-3503-3333-6/23/$31.00 ©2023 IEEE
DOI 10.1109/MetaCom57706.2023.00078

II. Related Works

The number of end devices, such as wearable devices, mobile phones, tablets, and Internet-of-Things (IoT) devices has grown explosively in the past decade. Many mobile and Internet of Things applications have become increasingly resource-intensive and latency-sensitive, such as web online gaming, Augmented Reality, and Autonomous vehicles. There exist several works [3][4][5] in the literature relevant to task offloading in MEC. However, they are centralized in nature which has few disadvantages. Firstly, centralized solutions are prone to a single point of failure. Moreover, it can be biased towards a few edge nodes while task offloading to make them earn more profits. As a result, recent years have witnessed research interest to utilize decentralized technologies such as a blockchain for task offloading because of its inherent characteristics of being trusted, highly available, and easily auditable. A few examples of these works are given below. In the work [6], the authors adopted a prospect theoretic approach with the Stackelberg game for blockchain mining tasks in MEC. By leveraging a Stackelberg theoretic approach, they formulated computing resource pricing and consumption as coupled leader-follower games where the different preferences of mobile device owners for competitive risks and profits were considered. The work [7] proposes a reinforcement learning (RL)-based dynamic task offloading scheme which enables Mobile Units(MUs) to make optimal offloading decisions based on blockchain transaction states, wireless channel qualities between MUs and MEC servers. To further improve the offloading performances for larger-scale blockchain scenarios, authors develop a deep RL algorithm by using a deep Q-network. A task offloading framework for edge computing based on consortium blockchain and distributed reinforcement learning is proposed in [8]. The authors introduced a framework that provides high-quality task offloading policies with data privacy. The framework consisted of three key components which are data quality evaluation (DQ), distributed reinforcement learning for task arrangement (DELTA), and data repairing (DR).

However, we have not found many works in the direction of peer-to-peer task offloading in MEC using blockchain to ensure mutual trust.

III. Proposed solution

This section is organized as follows. Section III-A narrates the system model and assumptions. Section III-B describes the process of task submission. The task bidding method is proposed in Section III-C. Section III-D deals with the task offloading process. Result Submission and validation method is discussed in Section III-E. Section III-F and Section III-G describe the reward payment and reputation management system respectively.

A. System model

The proposed framework is comprised of heterogeneous edge nodes. The system model has illustrated in Fig. 1. Edge nodes are from various manufacturers and are connected to the Internet. An edge node may act as a task offloader that requires computation offloading or may act as a computation resource to perform the offloaded task. The proposed MEC framework works with a reward system implemented by a digital token named "Computation Reward Token (CRT)" which is maintained by the proposed blockchain. Stakeholders interested to deploy a MEC network in a region can come up with edge devices and establish an overlayed peer-to-peer network among themselves. The proposed blockchain-based architecture is shown in Fig. 1. The major steps involved in our approach are enlisted as follows.

1) An edge device publishes a task offloading request.
2) Executor devices participate in the bidding procedure.
3) Task allocation through smart contracts.
4) Task execution and result submission by the executor.
5) Result verification and reward.
6) Reputation update of the executor node.
7) Finally, the smart contract is destroyed.

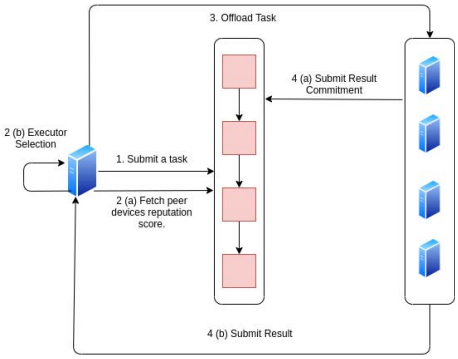

Fig. 2. Workflow of our approach

Fig. 2 shows the workflow of our approach. The detailed methodologies of each of the above-mentioned steps are given below.

B. Task publishing

When an edge node requires some computation offloading, the request is published through a new smart contract on the blockchain. All the nodes of the peer-to-peer network get aware of the smart contract and the offloaded task. The task contains a list of small sub-tasks to be performed. Each sub-task is defined in terms of computation, storage, and delay requirements.

C. Task bidding

The bidding procedure can be divided into two phases:-

1) *Bid Submission Phase*: In the bid submission phase a contender node submits a commitment for the amount of token that it wants to put on stake. The commitment is defined as $h = H(M)$ where $M = \{(node\ id)||(number\ of\ token\ on\ stake)\}$. All contending devices submit their commitments in the bid submission phase.

Algorithm 1 Creating preferred edge list

Require: L : List of devices that participated in the Bidding process, d_{ID} : ID of device d, d_R : Reputation score of device d, d_S : Number of tokens staked by device d, d_C : Number of subtasks the node is ready to take d, d_L : Latency value between device d and the offloading device, d_{SCORE} : Score of device d

Ensure: Return list S of devices to which the task shall be offloaded. S is sorted in decreasing order of the best suitable device.

START
while $L.length() \neq 0$ **do**
Pick a device d from L
d_{SCORE} = d_R * d_S * d_C * 1/d_L
S.push((d_{ID},d_{SCORE}))
$L.remove(d)$
end while
return S.sortDecending(2) {Sort S in non increasing order based on the second parameter i.e. d_{SCORE} }
END =0

2) *Bid Revelation Phase*: In the bid revelation phase, each contender submits or discloses the M to the smart contract satisfying the hash value h.

At the end of the bidding procedure, a sorted list of potential contenders is obtained. The list is sorted in non-increasing order of the score. The smart contract computes the fitness score of an edge node by combining its reputation, staked tokens, resources, and latency from the offloading device. The list is sorted based on Algorithm 1.

D. Task Offloading

The offloading edge node needs to make a trade-off between the cost and the redundancy. It can offer a high CRT amount as a reward to attract highly reputed edge nodes for bidding in which case the result from only one node is trusted. Alternatively, the offloading edge node can offer a low CRT amount for each computing node but increase the redundancy factor to have the computation done by multiple new nodes of the eco-system to improve the reliability of the result. The smart contract refers to the list of potential contenders and allocates the offloaded task. More than one resource node can be chosen based on the redundancy factor. The smart contract includes the hash of the data and the algorithm for the task or the sub-task. The data is transferred to the designated device through the carriers. The task offloading is performed as per the Algorithm 2 and is implemented as a smart contract function which is called by the offloading node. An edge node that has bidded for the offloaded task and has not received the task, gets back its staked token after the offloading device completes its task offloading.

E. Result Submission and validation

On the completion of the offloaded task, a computing edge device submits the result to the smart contract in the form of a hashed commitment in two phases of the submission and revelation phases so that no edge node can copy the result from others. The two phases are given below.

Algorithm 2 Task Offloading

S: Sorted list of edge devices.
r: Redundancy factor
START
1) For a task T from D_i obtain an ordered list S obtained from the Device Selection Algorithm (Algorithm 1) where for each $D_j \in$ S $Latency_{D_i}^{D_j} \leq \delta$, D_j is having sufficient reputation score, D_j has staked token.
2) If S is empty repeat step 1
3) Divide T into $t_1, t_2, t_3, ..., t_n$

while r ≥ 0 **do**
for each t$_i$ $\in$t$_1, t_2, t_3 ... t_n$ **do**
for each $D_j \in$ S **do**
if t_i not allocated to D_j and D_j has available resources **then**
Assign t_i to a D_j
break;
end if
end for
end for
r = r - 1
END =0

1) *Result submission phase*: In the result submission phase a node submits the hash of the result defined as $h = H(M)$ where $M = \{(node\ id) \| (result\ value)\}$
2) *Result revelation phase*: In the result revelation phase, each contender submits or discloses the M to the smart contract satisfying the hash value *h*.

F. Reward payments and punishments

If the assigned resource node performs the task correctly and submits correct results, it can execute the designated function of the smart contract to claim the CRT reward to be transferred from the smart contract account to its own account. Along with that, the task-executing device also claims its staked CRT back. So the total tokens being received will be reward CRT + staked CRT. However, if the computation performed by a peer device is found to be wrong no reward token is transferred to the computing device. Instead, the tokens are transferred to the offloading device to compensate for the loss incurred due to the non-availability of valid results. The reputation of the nodes is calculated by the smart contract accordingly as explained in Section III-G.

G. Reputation Management

Reputation Management is the core of our approach. The reputation score of an edge server determines the reliability of the result produced by an edge node. Depending on the reputation value of the devices we have categorized the devices into 3 categories as given below.

- Category A : Reputed, an edge node with a reputation score of 75 and above on a scale of 0-100. These devices are the highly trusted devices in the network based on the reputation score obtained from their performance so far in the ecosystem. Results obtained from such devices can be directly taken into consideration.
- Category B : Benign, an edge node with a reputation score between 40-74 both inclusive on the scale of 0-100. These devices are benign in nature which means they haven't performed any misbehavior yet based on the reputation score obtained from their performance so far in the network. But their reputation is not as high to be in Category A. An offloading device can rely on the computation done by a Category B device with a confidence probability. The c score is decided by the offloading device on factors such as result feedback from past interactions with the peer devices. c. The higher the c score lesser is the redundant verification performed.
- Category C: Undecidable, an edge node with a reputation score between 0-39 both inclusive on the scale of 0-100. These are those devices that have either recently joined the network or have a history of misbehaving. An offloading device shall not ideally trust the outcome of the computation done by Category C devices. Instead, the outcome needs to be further verified by redundant computations from other nodes available in the vicinity. A task t offloaded to a Category C edge devices d is validated by the most occurrence number of the result values submitted by the devices.

A new device joining the network join as a category C device, with a reputation score of 0. Over time the device participates in the task verification procedure and earns a higher reputation score. A node can belong to category C also due to the demotion from higher categories. The reputation of an edge device plays a crucial role in deciding whether an offloaded task will be allocated to it or not. A reputation management scheme must take into account the way the reputations are increased or decreased. Edge devices should be encouraged to maintain a consistently good reputation score throughout their lifetime instead of maintaining an inconsistent score. Our reputation scheme is developed on the ideology that, **It takes a long time to make a reputation and takes no time to lose it.** The scheme uniformly increases the reputation but strongly penalizes in case of misbehavior. Also, the correct execution of offloaded tasks from an edge node of different manufacturers brings more credits than the task offloaded by its own manufacturer.

The following cases are considered for reputation update for a device D_j which has performed an offloaded task t_i from the device D_i.

$Case1$**:** If D_j is not from the same manufacturer of D_i and result submitted for task t_i is correct.

$Case2$**:** If D_j is from the same manufacturer of D_i and result submitted for task t_i is correct.

$Case3$**:** If D_j is not from the same manufacturer of D_i and result submitted for task t_i is incorrect, or the result is not submitted.

$Case4$**:** If D_j is from the same manufacturer of D_i and result submitted for task t_i is incorrect, or the result is not submitted.

Symbol	Meaning
D	An Edge Device
T	A whole task
t_i	ith portion of T
d_T	Deadline of T
δ_T	Minimum Reputation for contending for T
$\text{Latency}_{D_i}^{D_j}$	Latency from $D_i to D_j$
p	Processing time for T by D_j
Δ	Reward for processing other Org. Task
CRT	Computation Reward Token
c	Confidence probability score of a result from a device

TABLE I
MEANINGS OF SYMBOL

$$D_{j,t_i}^{rep} = \begin{cases} Case1: & \text{(d/p) * } \Delta \\ Case2: & \text{(d/p)} \\ Case3: & \text{-2} * 1/(d/p) * \Delta \\ Case4: & \text{-1}/(d/p) \end{cases} \quad (1)$$

Similarly, the total reputation score of the device gets updated in the smart contract as,

$$D_j^{rep} = \frac{((D_j^{rep} + D_{j,t_i}^{rep}) - Min(D^{rep}))}{(Max(D^{rep}) - Min(D^{rep}))} * 100 \quad (2)$$

D_{j,t_i}^{rep} is the reputation score earned by device j for performing task t_i, t_i = T if task is not fractioned. D_j^{rep} is the reputation score of device j. $Max(D^{rep})$ is the maximum Reputation score in the system. $Min(D^{rep})$ is the minimum Reputation score in the system. The other symbol meaning is given in TABLE I. The logic increases the reputation score of a device D_j in the proportion of the earliness of its task finish time. Consecutively correct result performed by a device for the task offloaded from another manufacturer will bring more reputation. At the same time incorrect result performed by a device for the task offloaded from another manufacturer will penalize the device D_j much more strictly.

IV. PERFORMANCE EVALUATION

In this section, we first formulate the computation problem that we consider to be offloaded to the edge nodes. We then attempt to solve the computation problem under different scenarios and discuss the performance of the system.

Problem Definition : We consider the problem of finding a correct OTP (One Time Password) value for a pre-image. We consider an OTP to be of 8 digits where a digit can have a value ranging between 0-9 each. An edge device E_0 decides to offload the task of finding out the correct OTP for a given OTP digest. We assume that there are 9 other edge devices excluding the offloading device available in the vicinity of

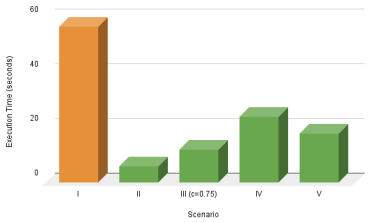

Fig. 3. Task completion time under different scenarios

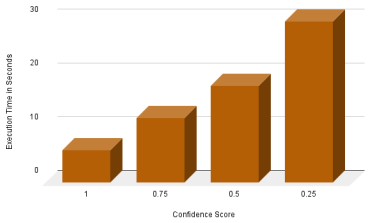

Fig. 4. Task completion time for Category B devices under different confidence score scenarios

the offloading device. Each device among the other 9 devices either belong to Category A, Category B or Category C.

We consider the following five scenarios under which the problem is to be solved:-

(I) When a task is not offloaded to other peer devices instead gets executed on the source device.
(II) When a task is distributed among 9 Category A devices.
(III) When a task is distributed among 9 Category B devices.
(IV) When a task is distributed among 9 Category C devices.
(V) When the edge devices include 1/3 of devices from each Category A, B and C .

For all the above scenarios the execution was made on a single CPU core. The procedure to find out the OTP value was written in python. Fig. 3 shows the execution time spent for performing a task under the five above-mentioned scenarios respectively. For scenario III we consider c=0.75. We divide a task equally among each device. Fig. 4 shows task execution time under scenario III with varying confidence scores ranging from 1 to 0.25.

We setup a permission instance of the Ethereum blockchain with 10 nodes. The blockchain was configured to run the Clique consensus algorithm, a Proof of Authority[9] based algorithm. The choice of the PoA algorithm is made because of its lightweight characteristics and is less energy-consuming than the challenge response-based PoW algorithm. Out of the 10 nodes 9 were designated as sealers or the validator and 1 was made as a Full Node. We execute a workload of 10000 transactions to check the performance of the approach. All the transactions were of type *result submission*. The transactions were executed from a non-validator node. The statistics can be found as below:-

Execution Time	62.212 seconds
Total No. of Blocks	63
Block Generation	1 second
Avg. Txns per block	159

From the above statistics, it can be seen that the Block Generation Time is set to be 1 second. It is an overhead for maintaining the blockchain system. We can justify the overhead as once the task gets offloaded to the peer devices rest of the execution can be performed faster as compared to the execution being performed without offloading the task. The claim is also evident from Fig. 3.

V. Conclusions

In this work, we proposed a novel framework based on permissioned blockchain for task offloading in Mobile Edge Computing. The smart contract protects honest offloaders from malicious nodes by validating the results and punishing them for wrong result submission. The smart contract also protects honest task performer nodes by preventing denial of reward payment from malicious offloader nodes. The decentralized system eliminates any possibility of biases in task offloading, and the monopoly over the eco-system. We have performed testbed experiments to find out the feasibility of our solution. From the experiments, we observed that offloading a task to peer devices the task completion time in comparison to when the task is executed on a single machine. Our choice for considering Proof-of-Authority as the consensus mechanism makes the blockchain system lightweight, thus allowing the edge device to spend less energy on the consensus mechanism.

References

[1] S. Dhelim, T. Kechadi, L. Chen, N. Aung, H. Ning, and L. Atzori, "Edge-enabled metaverse: The convergence of metaverse and mobile edge computing," *arXiv preprint arXiv:2205.02764*, 2022.
[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.
[3] F. Wei, S. Chen, and W. Zou, "A greedy algorithm for task offloading in mobile edge computing system," *China Communications*, vol. 15, no. 11, pp. 149–157, 2018.
[4] G. Li and J. Cai, "An online incentive mechanism for collaborative task offloading in mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 624–636, 2019.
[5] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7635–7647, 2019.
[6] K. Zhang, J. Cao, S. Leng, C. Shao, and Y. Zhang, "Mining task offloading in mobile edge computing empowered blockchain," in *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*. IEEE, 2019, pp. 234–239.
[7] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2536–2549, 2020.
[8] Y. Liu, X. Guan, Y. Peng, H. Chen, T. Ohtsuki, and Z. Han, "Blockchain-based task offloading for edge computing on low-quality data via distributed learning in the internet of energy," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 2, pp. 657–676, 2021.
[9] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain," 2018.