Cloud Computing (CS466)

Blockchain-Based Collaborative Task Offloading in MEC: A Hyperledger Fabric Framework

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, SURATHKAL

Submitted By :
Sayad Shahanaz - 181CO247
Mudavath Prathyusha - 181CO233
Afrah nayeem - 181CO142

Abstract:

While task offloading is one of the most critical issues in Mobile Edge Computing systems, it is frequently approached entirely from the end-user's standpoint. Traditional MEC frameworks overlook the possibility of task offloading across edge servers, enhancing overall utilisation and

efficiency. Task offloading services may be vulnerable to security and privacy issues. Blockchain can be used to ease the deployment of a task offloading scheme in a MEC architecture. The proposed mechanism is based on the Hyperledger Fabric blockchain, which provides a lightweight distributed interaction field that solves various security and privacy issues. The main aim is to propose a blockchain-based Service Oriented Architecture for safe task offloading coordination among MEC edge servers.

Introduction:

Mobile Edge Computing (MEC) is a network architecture that relocates cloud services such as computing, storage, and control functions close to end-users. The main idea behind MEC is that the overall performance and associated quality-of-service (QoS) of a communication network can be improved by distributing servers, i.e. edge servers, spatially close to end-users. In an actual deployment, an edge server is strategically placed to serve a geographical area. Upon receiving computation job requests (e.g. tasks) by end-users, the server must process them out without the possibility to refuse them access to its premises. However, in a client-dense setting, this could overwhelm the server's resources, resulting in a situation where clients' performance expectations cannot be met. In traditional frameworks, the cloud level is still responsible for handling this problem by being in charge of finding available resources within its watch and then moving the tasks from a busy server to an available one for the appropriate processing. This, as expected, could also harm the network's performance and deteriorate the QoS for end-users. In addition to that, the existing literature on MEC frameworks mostly approach the task offloading problem from the wireless/mobile end-user point of view. They do not account for the scenario where tasks can be shared among servers at the edge level, even though a collaboration mechanism would improve the performance of traditional MEC frameworks. However, such architecture might still be vulnerable to many security and privacy threats that need to be addressed.

The proposed solution will run on top of Hyperledger Fabric as a provider of a lightweight distributed interaction playing field for the edge servers that minimises many security and privacy concerns. The main goal is to put forward a Service Oriented Architecture (SOA) that allows secure and private task offloading among edge servers to help alleviate processing saturation in dense networks.

The main contributions can be summarised as follows:

- Proposal of a system model for the Hyperledger Fabric-based task offloading collaboration scheme for edge servers in a MEC environment.
- Presentation of a solution mechanism.

- Discussion of preliminary simulation results to evaluate the viability of the proposed system.

Hyperledger Fabric BlockChain

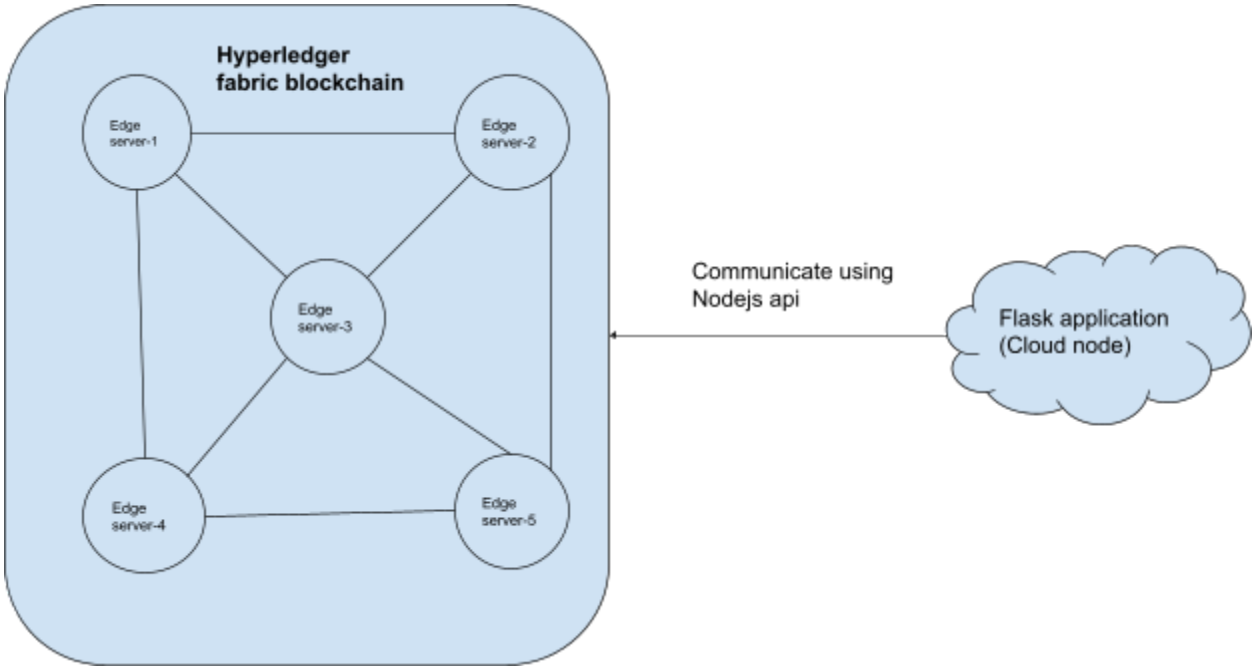
Hyperledger Fabric is a permissioned blockchain that provides a secure, private and scalable platform for implementing multi-purpose distributed applications. Its modular and versatile design satisfies a broad range of industry use cases. Hyperledger Fabric allows consensus and membership services to be plug-and-play. The fabric also implements the idea of customisable intelligent contracts, also known as chain codes, to govern the interactions among participants in the framework.

The typical transaction flow in a Fabric framework can be divided into four phases: proposal, endorsement, ordering, and commitment. The flow begins when a client node sends a transaction proposal to one or more peers in the network, hoping to collect endorsements for the transaction. The peers simulate the transaction and endorse it. After acceptance, the transaction is sent back to the client, who then submits it to the blockchain's ordering service. The ordering service then implements the consensus protocol before submitting the transaction to a committing peer for final validation before being included in the ledger. There are numerous Fabric configuration parameters, both global and peer-specific, that can affect the framework's performance as measured by transaction latency and throughput.

However, the parameter space is vast, and it could involve variables like endorsement policy, block size, ledger type, consensus type, etc. Endorsement policy, for example, defines how many times a transaction must be executed before an order request can be forwarded to the ordering service and thus directly impacts the blockchain's latency. Another example is block size, which specifies the maximum number of transactions that can be recorded in a single block. As most of the verification steps are conducted per block, block size impacts the system's latency and throughput. We'll employ some of the configuration parameters that significantly impact the fabric performance during the task offloading mechanism.

PROPOSED SOLUTION MECHANISM

The proposed mechanism leverages the management technology of Hyperledger Fabric along with Mobile Edge Computing to provide a scalable platform to oversee edge computing servers.



Hyperledger Fabric-based task offloading architecture for MEC servers.

The above figure illustrates the mechanism. First, the Fabric is a permissioned blockchain, meaning there is strict access control enforcement to the blockchainMEC environment. This feature prevents malicious edge-server actors with pernicious motives to join the network.

Point-to-point communication among nodes is allowed by the fabric; also, every node keeps a ledger with different states. In fact, the ledgers kept by the nodes are different in the sense that only nodes affiliated with a specific transaction can access the shared state and content of the ledger. This feature enhances the confidentiality of the mechanism.

Consensus service is lightweight; it uses smart contracts and a notary service to validate inputs, outputs, transactions, identities, and digital signatures. This feature preserves the idea of a distributed and intelligent collaborative system at a low computation cost.

These ideas are summarised in the list down below :

1. Service Request: A busy ordering node owns a computation job with a set of tasks; the ordering node requests the offloading service to the cloud node.

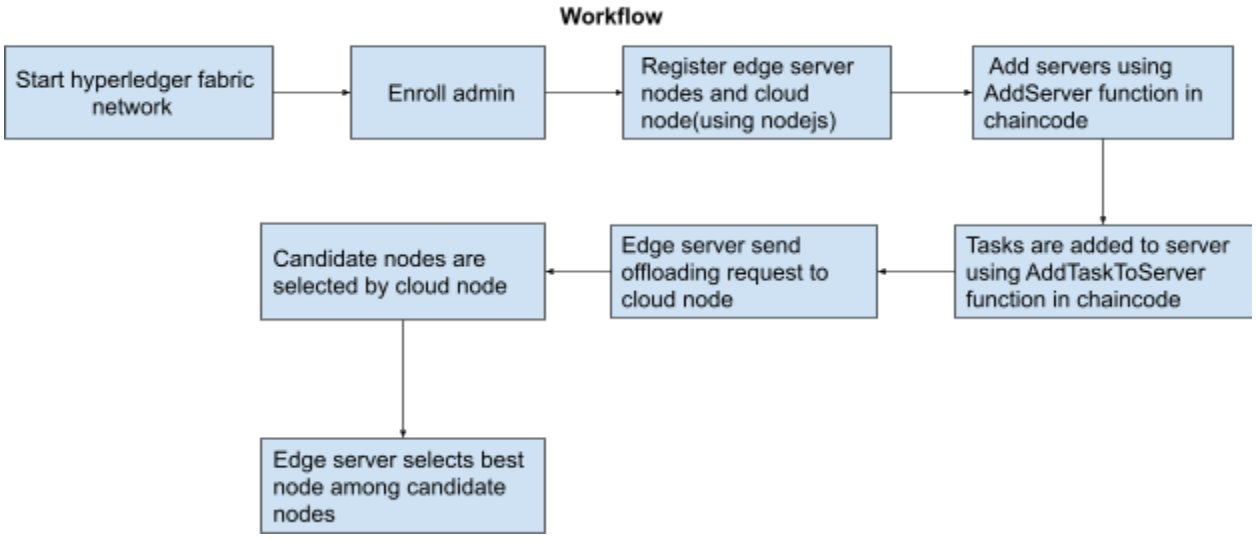
2) Discovery of candidate servers: The cloud node runs a monitor tool with an overview of the system that includes the available resources in the network and the Fabric performance metrics. The cloud node shortlists a set of candidate servers to offer the offloading service. It passes on

the list of best candidate nodes to the ordering node so it can decide on the optimal offloading decision.

3) Offloading Decision: The ordering node locally solves an optimisation problem to find the optimal allocation of tasks that maximises its cumulative utility. Once the offloading decision has been made, the ordering node communicates it back to the cloud, which subsequently passes it on to the candidate selected as a service provider.

4) Task uploading, remote execution, and return of results: Each service provider and the ordering node start an ISC to upload, execute, and return the results of the offloading service. The ISC includes many stages, such as the verification of membership credentials, endorsement of the offloading transactions, consensus mechanism, and commit process to the ledger. This flow will follow the Fabric framework by establishing a peer to peer communication between the involved parties preserving the security and privacy of the offloading mechanism.

The service-oriented architecture (SOA) presented in this work provides a smart, distributed, and model-driven communication frame for servers in a MEC environment. The architecture supports a flexible, secure, and lightweight task-sharing framework with the ultimate goal to increase the utilisation of computing resources in a MEC environment.



Smart contract functions:

1. AddServer - Adds server (attributes- server_number, cpu_clock_frequency, hardware_value, computation_power)

2. QueryServer - We can get the server information by providing the server number
3. QueryAllServers - Using this function, we can get information about all servers.
4. AddTaskToServer - This adds task to each server (Task attributes - task_size, task_number,deadline,cpu_cycles)

Task Offloading Model At The Server Level

Consider a group of M edge servers and a cloud, all of which are running on the Hyperledger Fabric blockchain as distinct nodes. Fabric will allow edge server nodes to interact without the requirement for a central authority to make offloading decisions. At the same time, Fabric will ensure that all valid offloading interactions are recorded in an immutable ledger, providing transparency, security, and trust.

Consider an ordering edge server node $m \in M$ with a CPU-clock frequency of FM. The ordering server m owns a computational job that is made up of a set of indivisible tasks. The tuple
(d_a , ρ_a , τ_a) represents a job $a \in A$, where d_a is the task size in bytes, ρ_a is the number of CPU cycles required for task execution, and τ_a is the task time deadline. At any given time, the server node m is trying to determine the optimal decision for set A. The ordering server has to decide whether to run the tasks locally or offload them to other servers in the system. If 'm' were to finish a job $a \in A$ locally, the local execution time and energy consumption would be estimated as follows:

$$\mathbf{t^a_m = \rho_a / f_m}$$
$$\mathbf{e^a_m = c_m f_m^2 \rho_a}$$

According to the DVFS computing model for MEC servers, c_m is a constant linked to the hardware design of m. If m is low on resources and wants to consider outsourcing the task, it would request to the cloud for the Hyperledger Fabric-based offloading service. The cloud node has an overview of the available resources of all the edge servers in the network. It can also query various performance metrics of the blockchain by using a monitoring tool.

When an ordering server requests offloading service, the cloud shortlists a set of candidate servers $N = \{1,..., N\}$, $N \subset M$ having adequate resources to provide the service. If the ordering server m offloads a task $a \in A$ to a candidate server $n \in N$ with f_n CPU-clock frequency, the offloading time and energy consumption would be as follows:

$$\mathbf{t_a^{mn} = (d_a / R) + (\rho_a / f_n) + t_{HFB}}$$

$$\mathbf{e_a^{mn} = (d_a / R) * P_m}$$

All nodes are assumed to be connected by some medium with data rate R. The task transmission time from m to n, the task execution time in n, and the delay introduced by the Hyperledger blockchain t_{HFB} make up the offloading time t^{a}_{mn}. The energy consumption e^{a}_{mn} during the offloading process is the task transmission time from m to n multiplied by the average power of the ordering server P_m. The time and energy-saving utility ratios can be computed as follows:

$$\varphi^{t}_{mn}(a) = (t_m^a - t_{mn}^a) / t_m^a$$
$$\varphi^{e}_{mn}(a) = (e_m^a - e_{mn}^a) / e_{mn}^a$$

The function $\Phi_{mn}(a)$ defines the utility gained by an ordering server node m if it offloads a task to a candidate node n. It quantifies the benefits of task offloading to the ordering server. $\Phi_{mn}(a)$ is the weighted average of four ratios, time-saving utility, energy-saving utility, endorsed transaction success rate, and normalised block size. The endorsed transaction success rate φ^{ε} denotes the ratio of accepted transactions to the total number of transactions submitted by the node to the blockchain. The normalised block size ratio φ^{β} is the ratio of the size of a block used for blockchain transactions to the maximum extent allowed by the Hyperledger Fabric.

$$\Phi_{mn}(a) = (\varphi^{t}_{mn}(a) , \varphi^{e}_{mn}(a) , \varphi_m^{\varepsilon} , \varphi_m^{\beta})$$

The ordering node's preferences for a specific utility ratio are expressed as weights. The total utility is the inner product of the utility ratios and the corresponding weights.

$$\Phi_{mn}(a) = w_m^{t} * \varphi_{mn}^{t}(a) + w_m^{e} * \varphi_{mn}^{e}(a) + w_m^{\varepsilon} * \varphi_{mn}^{\varepsilon}(a) + w_m^{\beta} * \varphi_{mn}(a)^{\beta}$$

Problem Formulation:

The offloading decision problem of an ordering server m having a set of tasks A that will be offloaded to a set of candidate nodes N is formulated as a binary integer linear program that maximizes the cumulative utility of the offloading service with respect to the decision variables x^{a}_{mn}.

$x^{a}_{mn} = 1$, if 'm' offloads task 'a' to 'n'
$x^{a}_{mn} = 0$, if 'm' executes it locally
MAX $\sum_{a \in A} \sum_{n \in N} (\Phi_{mn}(a) * x^{a}_{mn})$ such that,
$x^{a}_{mn} \in \{ 0 , 1 \}$
$\sum_{n \in N} x^{a}_{mn} = 1$

$t^{a}_{mn} \; x^{a}_{mn} < t^{a}_{m}$
$e^{a}_{mn} \; x^{a}_{mn} < e^{a}_{m}$

$$t^{a}_{mn} \; x^{a}_{mn} < \tau_{a}$$
$$\eta^{a}_{n} \; x^{a}_{mn} < 1$$

Constraint 1 indicates that the decision variables of the problem are binary. Constraint 2 ensures that a task can be allocated to at most one candidate. Constraints 3 and 4 ensure that only the candidate nodes offering better processing times and energy savings are selected. Constraint 5 guarantees that the offloading decision meets time deadlines of all tasks. Finally, Constraint 6 prevents offloading tasks with CPU needs greater than the available CPU resources to the candidate nodes.

After an offloading decision is made, the ordering node m and the service provider n will start an Installed Smart Contract supported by Fabric. The ISC will follow the Fabric logic framework in terms of transaction proposal, endorsement, validation and committing the offloading transactions to the blockchain ledger.

Pseudocode for HFB based MEC offloading routine

for each m $\in$ M = {1,...,M} do
    m $\in$ M owns a computation job with a set of tasks A;
    m requests offloading service to the cloud;
    the cloud shortlists a set N of candidate servers;
    for each a $\in$ A = {1,...,A} do
        for each n $\in$ N = {1,...,N} do
            compute $\Phi_{mn}(a)$;
    find optimal offloading decision by solving the ILP assignment problem;
    pass offloading decision to the Cloud;
    initiate offloading ISC following HFB framework;
    return offloading results;

Conclusion:

A blockchain-based collaborative task offloading mechanism for edge computing servers is developed to increase the utilization of the available resources at the edge layer. The method is mainly based on the Hyperledger-Fabric blockchain, which provides a distributed interaction platform with powerful cryptographic features to reduce the risk of security and privacy breaches. The offloading technique is defined as an integer linear programming problem with binary decision variables and fabric performance measures. Our work includes an implementation of the offloading mechanism on the live Hyperledger Fabric blockchain network.