A Study of Edge Computing Offloading Based on Security

Luo Pei
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710021, China
E-mail: 1985815398@qq.com

Lei Juchao
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710021, China
E-mail: 274766943@qq.com

***Abstract*—With the widespread use of IOT scenario and the development of mobile network technology, the access to many devices at the edge of the network has generated an exponential increase in the volume of data, bringing higher than ever before the demand for data transmission bandwidth, traditional centralized cloud computing can no longer meet the demand, the need to adopt edge computing distributed collaborative approach to data processing. In this paper, we study a mobile edge computing model based on energy consumption optimization management under a certain delay constraint, focus on an edge computing offload scheme based on security management, and design an offload decision scheme based on a multi-objective optimization hybrid quantum evolution algorithm (MHQEA) to ensure the security of user equipments during computing offload in the edge network and reduce total system energy consumption.**

Keywords-Edge Computing; Computing Offloading; Resource Allocation

I. INTRODUCTION

In recent years, with the rapid emergence of compute-intensive applications such as high-definition video, augmented reality/virtual reality, Internet of Things, Internet of Vehicles, and industrial Internet, people have put forward higher demands on the transmission rate and service quality of the network, resulting in an explosive growth in network traffic. At the same time, mobile devices are gradually emerging, due to the increasingly powerful design features, mobile devices have limited computing power, poor real-time and energy consumption limitations, it is difficult to bear the needs of computation-intensive, latency-sensitive and high energy consumption applications, In order to meet these challenges, the first solution proposed is to apply compute offload technology in mobile cloud computing (Mobile Cloud Computing (MCC)) architecture. By offloading the compute tasks of the terminal to a cloud data center with sufficient compute and storage resources, the delay and power consumption problems caused by the lack of computing power of the smart terminal can be alleviated to some extent.

In order to solve the latency problem of MCC, researchers have focused on mobile edge computing, which is a core technology of 5G in 2014[2], it was proposed by the European Telecommunications Standardization Institute (ETSI) to provide computing, communication and storage capabilities closer to users by sinking cloud data centers to the wireless network edge. In mobile edge computing (MEC) systems, network edge devices such as base stations, access points and routers are given cloud-like computing and storage capabilities to serve users as an alternative to the cloud. At the same time, because it is placed close to mobile device terminals and data sources, it can significantly reduce mobile device load and reduce transmission latency. Among them, compute offloading, as a cutting-edge technology for edge computing, can be achieved by offloading compute tasks to a well-resourced edge cloud. The problem of computational offloading is mainly focused on offloading decisions and resource allocation, and many scholars have done relevant research on these issues. in the literature[3], a one-dimensional search algorithm is used to minimize the latency target and optimize the user task execution latency based on the application queue buffer queue state. Experimental results show that the optimal strategy proposed in this scheme can reduce the latency by up to 80% compared to the local execution strategy and up to 44% compared to the cloud execution strategy, which can effectively respond to the dynamic arrival of intensive applications; the user device in the literature[4] adopts energy harvesting technique to optimize the energy consumption of the execution task, and proposes that the dynamic optimization algorithm can reduce the execution time by up to 64% by offloading the task to the MEC, while preventing the offloader from being dropped. The shortcomings of both papers are that they do not consider the energy

consumption of the user device, in the literature[4], it is assumed that the user device utilizes energy harvesting techniques and the energy consumption of the user device is ignored in the decision making process, energy harvesting techniques do not completely solve the energy consumption problem. The literature[5] investigates offload decisions during single node computation, the goal of which is not only to minimize execution latency but also to minimize the power consumption of edge computation, considering dense user devices capable of accessing multiple edge servers through nearby small nodes, and then proposes an optimal solution using an equivalent discrete framework, however, as the number of edge servers increases, this approach leads to high communication overhead and high computational complexity. Therefore, the authors solve this problem through an application assignment indexing scheme, which broadcasts through the node's own indexing policy and selects the most appropriate edge server to minimize execution latency and energy consumption, the downside of this scheme is that it does not consider scenarios with multiple compute nodes. The literature[6] considers the problem of joint optimization of energy consumption and latency of devices in a multi-user, multi-channel environment, where devices can optimize performance by adjusting the weighting parameters, however, this literature assumes that the computational resources of the MEC are sufficient, and the problem of insufficient edge computing resources exists in realistic production environments with multi-user conditions. However, the above literature does not consider the impact of security issues on system performance. Security issues such as privacy breaches may be encountered during the offloading process, so security issues are equally important in the study of edge computing offloading.

Unlike the aforementioned literature, this paper designs a secure computational offloading method, the main contents of which are as follows: considering the impact of time delay and energy consumption on the system, the resource allocation in the task transfer process and the offloading decision in the task processing process are jointly optimized to achieve the cost optimization of mobile devices, and security is added on top of this. The specific work is as follows:

1) Design the MEC network architecture and ensure security.

2) Considering minimum energy consumption under time constraints and joint optimization to improve QoE.

3) An offloading decision solving method based on the Multi-Objective Optimization Hybrid Quantum Evolution Algorithm (MHQEA) is proposed.

4) Through simulation and comparison with conventional algorithms, it is proved that the energy saving cost proposed in this paper is lower, the total energy consumption of the system is lower, and the safety of the system is guaranteed.

II. SECURITY-BASED MEC CALCULATION OFFLOADING SCHEME

The compute offload technology offloads the compute tasks of the user terminal to the cloud service, solving the bottleneck in compute performance, resource storage, etc. of the user terminal. Computing offload technology was first proposed in the cloud, and the emergence of MEC has provided a new direction for the development of computing offload technology. Performing computational offload tasks in MEC not only solves the problem of high network resource utilization, but also solves the problem of latency. Therefore, the convergence of EMC and compute offload is an important direction for network development in the future.

In mobile edge computing, the edge cloud needs to process tasks from a variety of mobile terminals in real time, and needs to coordinate the distribution of edge server resources and task loads to meet the heterogeneous requirements of different tasks for processing delay, execution energy consumption and reliability. Offload decision and resource allocation will directly affect the performance of computing offload in mobile edge computing, which has great research significance.

A. Security-based system model design

Although edge computing networks reduce request latency and core network pressure and improve network performance, but some problems in network security have been exposed, especially its distributed deployment mode leads to network security reduction, making DOS attacks easier, therefore, how to ensure the security of edge computing becomes one of the problems facing edge computing networks.

The system establishes a model of multi-user multi-edge computing services in a wireless network. It is assumed that each edge server has complete control over the channel gain, local calculation energy, and input data size of all end users. In daily life, the number of users is greater than the number of base stations set in the edge cloud. We assume that the MEC system

consists of m mobile users and n base stations that can be connected to the edge cloud.

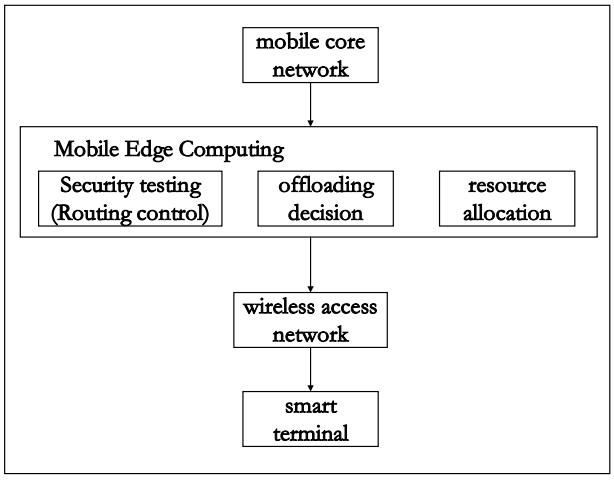

Figure 1. Security-based mobile edge computing network architecture

B. Design of Unloading Scheme for Edge Computing Based on Security

The computational offloading technique in edge computing involves the offloading decision and resource allocation, etc. The offloading decision is optimized for the performance of different services, and in this paper, the offloading decision is carried out with minimum energy consumption under the delay constraint, and the resource allocation is considered after the decision is offloaded.

1) Network model

We use C={c1,c2,... ,cn} to represent the different edge nodes, and the MEC set of each edge node is represented as M={m1,m2,... .mp}, which provides computational offload for mobile terminals, and the UE set is denoted as N={n1,n2,... .nq}.

2) Offload decision model

The offload decision model, as the core issue of computational offload techniques, is largely dependent on the computational task through the computing power of the device itself and the time delay and energy consumption that results when the offloading is completed to the edge. Therefore, it is necessary to calculate and analyze the delay and energy consumed by local computing and edge computing to complete the computing task.

The unloading decisions within each time slot are expressed in matrix form, i.e., the unloading decision matrix is represented by A, as in (1).

$$A=\begin{bmatrix} a_{11} & \dots & a_{1,m} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{n,m} \end{bmatrix} \tag{1}$$

Where, $a_{n,m} \in \{0, 1\}$ denotes whether device n offloads the task to the edge server, $a_{n,m}$ =0 denotes local execution, and anyway denotes the task is offloaded to the edge server. Each edge server can only handle one task at a time. Thus the constraints are as in (2).

$$\sum_{n=1}^{N} \mathrm{a}_{n,m} \le 1, \sum_{m=1}^{M} \mathrm{a}_{n,m} \le 1 \tag{2}$$

3) Local execution

Local execution means that user hands over task t to the mobile device for direct execution and the local execution of the task consumes mainly computational delay, as in (3).

$$T_{\mathrm{n}_i}^{l} = \frac{C_{n_i}}{f_{n_i}^{l}} \tag{3}$$

where C_{n_i} represents the CPU required to complete the task and $f_{n_i}^{l}$ represents the computing power of user n's mobile device.

The energy consumption for the local execution of task t is mainly the energy consumption for processing computational tasks, as in (4).

$$E_{\mathrm{n}_i}^{l} = \alpha_{\mathrm{n}_i} \cdot C_{n_i} \tag{4}$$

In the formula, α_{n_i} indicates the energy consumption factor of the mobile device.

According to formula (3) and formula (4), the locally executed valuation function is obtained as in (5).

$$\begin{aligned} & D_{\mathrm{n}_i}^{l} = \beta_{n_i}^{l} T_{n_i}^{l} + \beta_{\mathrm{n}_i}^{e} E_{n_i}^{l} \\ & s.t. \quad \beta_{n_i}^{l}, \beta_{n_i}^{e} \in [0,1], \beta_{n_i}^{l} + \beta_{n_i}^{e} = 1 \end{aligned} \tag{5}$$

In the formula, $\beta_{n_i}^{l}, \beta_{n_i}^{e}$ used as the weight of the local execution time and energy consumption of the

task, respectively, in order to determine the energy consumption of each user.

4) MEC server implementation

The execution mode of the MEC server refers to that the user n migrates the task t to a virtual machine on the MEC server through a wireless channel for execution, and allocates computing and communication resources to the user equipment through the trust detection server. The delay of the entire calculation task is mainly transmission delay and calculation delay, as in (6).

$$T_{\mathrm{n}_i}^{c} = \frac{B_{\mathrm{i}}}{R_i} + \frac{H_{\mathrm{i}}}{F} \tag{6}$$

The data volume of the calculation result is much lower than the data input volume, so the transmission consumption of the calculation result is ignored. The energy consumption of the task t server mainly includes transmission energy consumption, MEC server calculation energy consumption and server monitoring energy consumption, as in (7).

$$E_{n_i}^{c} = P_{n_i} \cdot \frac{B_{\mathrm{i}}}{R_i} + \gamma H_{\mathrm{i}} + E_{monitor} \tag{7}$$

According to formula (6) and formula (7), the locally executed valuation function is obtained as (8):

$$D_{n_{\mathrm{i}}}^{c} = \beta_{\mathrm{n}_i}^{c} T_{n_i}^{c} + \left(1 - \beta_{n_i}^{c}\right) E_{n_i}^{c} \tag{8}$$

5) Security inspection

Introduce a security module in the MEC server to perform security checks on the offloading process. Because the routing module is responsible for the traffic, and because edge nodes have limited capacity compared to cloud computing centers, they are vulnerable to traffic attacks, and although individual edge nodes are compromised and nearby networks quickly find replaceable nodes to adjust to, a malicious attack can bring down a server. Therefore, traffic forwarding is divided into traffic types and firewalls are set up between the center and branches to enhance DDoS protection. It also detects virtual machine operation in real time to prevent malicious virtual machine migration behavior.

The security detection module mainly detects the offloaded computational tasks through entropy detection algorithm, which can detect anomalies sensitive to the information entropy of changes in network parameters and UE parameters. In the case of potentially malicious offloads, the MEC controller transfers them to the security monitoring server, updates their trust values, labels them as trust violations, and verifies the user's access to the network through a combination of trust violation values and information entropy, increasing detection accuracy and reducing system overhead by unifying the monitoring of potentially malicious users. If there is any error in security detection, the defense policy module will continue to secure the network through reallocation of resources or security transfer.

The entropy detection algorithm can accurately sense changes in network parameters and calculate the corresponding information entropy to detect whether a user is maliciously unloading. Energy is consumed during the detection of offloading computational tasks, aiming at the minimum energy consumption under time constraints and based on a safe offloading scheme, where the attributes involved in the entropy detection algorithm are user trust, offloading frequency, network environment, CPU and memory utilization, etc. The distribution of the property z in G belongs to a polynomial distribution, and the probability equation is as in (9).

$$P(G_{\mathrm{z}}) = \frac{|G|}{\Pi_{\mathrm{z}=1}^{|G|} z!} \Pi_{z=1}^{|G|} G_{\mathrm{z}} \tag{9}$$

Where, the set of attributes G={g1,g2, ... ,g3}($1 \leqslant z \leqslant 5$), A represents the proportion of users with attribute G_{z} to the entire system users, $R_{n,i} = \sum_{G} P$ can be calculated, we use the maximum threshold method to determine if there is an abnormal packet unloading, if it is greater than the set threshold, then it is a malicious unloading task, unloading is not allowed during the unloading decision. The entropy detection algorithm is as in (10).

$$R_{\mathrm{n}} = -\sum_{i=1}^{5} R_{n,i} \log(R_{n,i}) \tag{10}$$

Where R_{n}^{H} is the detection threshold and unloading is determined by the detection result, as in (11).

$$\delta_n = \begin{cases} 1, & R_n \geq R_n^H \\ 0, & R_n < R_n^H \end{cases} \quad (11)$$

Detection of controlled energy consumption as in (12).

$$E_{\text{monitor}} = \frac{M_n R_n}{M_c} \quad (12)$$

E_{monitor} represents the server's monitored energy consumption, M_n represents the memory resources provided by the MEC server for one user device, and M_c represents the resources available for the entire server.

The overall optimization function is expressed as in (13):

$$\min[\sum_{n\in N}\left(\sum_{c\in C}\left(\delta_n E_{n_i}^l\right)+\left(1-\delta_n\right)E_{n_i}^c\right)]$$

$$s.t.\ \ \delta_n = \begin{cases} 1, & R_n \geq R_n^H \\ 0, & R_n < R_n^H \end{cases} \quad (13)$$

$$\sum_{n\in N}\delta_n F_{n,c} \leq F_c$$

In the constraint condition, F represents the total CPU computing resources of the MEC server. The computing resources allocated by the MEC cannot exceed the total computing resources.

III. Algorithm Designs

In the security-based computing offload scenario, the problem is an NP-hard problem because of the large data volume. To further solve this problem, this paper adopts a solution of MHQEA to find an optimal approximation of the model, and the process of finding an optimal approximation is represented as Algorithm 1.

IV. Simulation Performance Evaluation

Set the system time gap length to 20ms, calculate the local calculation energy consumption and unloading energy consumption by the total number of system user equipments 10, 20, 30, 40, 50, 60, respectively, and repeat each set of calculation 1000 times to take the average. As shown in Figure 2.

Algorithm 1: A computational offloading algorithm based on MHQEA

```
begin
    The current iteration number t = 0, and the maximum iteration
number is set to M
    Initialize the offload decision matrix An(t)
    Find the optimal solution matrix Pn(t) by observing the state of
An(t) through the make subroutine
    Correction of Pn(t) by reparation subroutine
    Evaluate the total energy consumption of the system and find the
optimal solution B(t)
      while(t<M)do
      begin
      Number of current iterations t plus 1
      An(t-1) is observed by the make subalgorithm to determine the
Pn(t)
      Evaluate Pn(t) for the minimum total system energy consumption
      Updating An(t) through the update subroutine
      Find the optimal solution in Pn(t) and B(t-1) to b
      if(Current number of iterations meets migration conditions)then
        Migration b to B(t)
        end if
    end
end
```

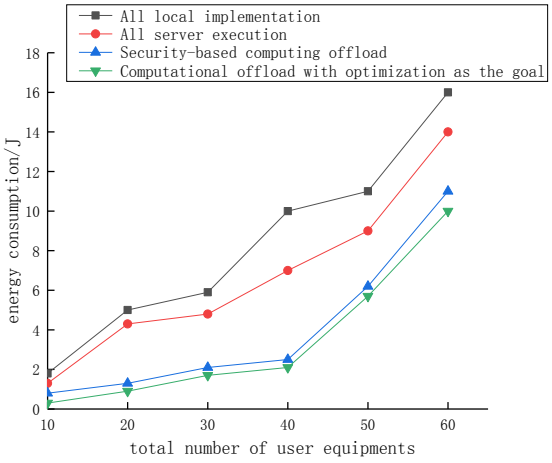

Figure 2. Relationship between offloading delay and total number of user equipments in different ways

As can be seen from Figure 2, the system energy consumption is higher for all local execution and all server execution, and the energy consumption is slightly higher for a security-based offload decision scheme than for an optimized energy-based offload

scheme. Because of the energy consumption required for security testing, it improves the overall security of the system and ensures the quality of service for the user's device unloading.

The optimal energy-based delay and security-based delay are calculated using the total number of user equipments in the system 10, 20, 30, 40, 50 and 60, respectively, and the average of each group is repeated 1000 times. As shown in Figure 3.

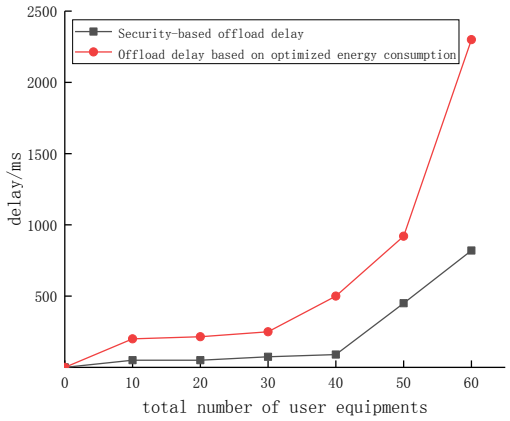

Figure 3. Relationship between different ways of offloading energy consumption and total number of user equipments

As can be seen from Figure 3, a security-based offloading decision scheme is better than an energy-optimized offloading scheme.

V. CONCLUSION

In the article, an offloading scheme for edge computing under security is proposed, which minimizes the total overhead of the system and improves the security of the offloading process, and we use an MHQEA-based algorithm to find the optimal offloading decision matrix. The simulation results show that the total system overhead for this option is lower than for other options, while maintaining safety.

Because MEC servers are small data centers, each server has far less energy than a cloud data center. With the development of the industrial Internet, MEC servers will be heavily deployed, which will cause the overall energy consumption of the system, how to deal with the greater energy consumption is also something we should study in the future.

REFERENCES

[1] Xie Renchao, Huang Tao, Yang Fan, Liu Yunjie. Principles and practice of edge computing [M]. Beijing: People's Publishing House, 2019.152-155

[2] Taleb T, Samdanis K, Mada B , et al. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration[J]. Communications Surveys & Tutorials, IEEE, 2017, 19(3):1657-1681.

[3] Liu J, Mao Y, Zhang J, et al. Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing Systems[J]. 2016.

[4] Mao Y, Zhang J, Letaief K B. Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices[J]. IEEE Journal on Selected Areas in Communications, 2016:1-1.

[5] Guo X, Singh R, Zhao T, et al. An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems[C]//ICC 2016 - 2016 IEEE International Conference on Communications. IEEE, 2016.

[6] Chen X, Jiao L, Li W, et al. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing[J].IEEE/ACM Transactions on Networking, 2015:1-1.

[7] Chen Yanbing, Li Juan, Zhang Qi. Exploration of quantum evolution algorithms for solving constrained multi-objective optimization problems[J]. Electronic Technology and Software Engineering, 2015, 000(016):P.185-186,187.