IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 20, 2019, accepted January 2, 2020, date of publication January 7, 2020, date of current version January 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2964626

Grouping-Based Consistency Protocol Design for End-Edge-Cloud Hierarchical Storage System

SHUSHI GU[1,2], (Member, IEEE), YIZHEN WANG[1], YE WANG[1,2], (Member, IEEE), QINYU ZHANG[1,2], (Senior Member, IEEE), AND XUE QIN[3]

[1]School of Electronic and Information Engineering, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China
[2]Peng Cheng Laboratory, Shenzhen 518052, China
[3]Department of Computing Sciences, Texas A&M University at Corpus Christi, Corpus Christi, TX 78412, USA

Corresponding author: Ye Wang (wangye83@hit.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61701136, Grant 61831008, and Grant 61525103, in part by the Guangdong Science and Technology Planning Project under Grant 2018B030322004, in part by The Verification Platform of Multi-Tier Coverage Communication Network for Oceans under Grant PCL2018KP002, and in part by the Shenzhen Basic Research Program under Grant JCYJ20170811154233370 and Grant ZDSYS201707280903305.

ABSTRACT With the increasing of the number of edge-devices and the demand of real-time experience, the end-edge-cloud hierarchical storage system (EECHSS) is emerged recently for reliable caching and fast offloading of massive amounts of data. EECHSS can accommodate various services, save computing power and improve storage capacity, due to transformation from central-cloud to edge-cloud, considerable reducing service delay and communication overhead. One of the main challenges brought by edge-cloud architecture is consistency problem. It is difficult to guarantee that the data from the two distributed clusters is consistent by existing consistency protocols. In addition, as the variety of applications increases, the existing fixed consistency level of classical protocols can no longer satisfy the system dynamic requirements. In this paper, we focus on designing grouping-based consistency protocols with adaptively selecting consistency level in EECHSS. At first, we analyze the internal structure and the workflow of EECHSS, and devise two modified adaptive grouping-based consistency protocols (GM-Paxos and GEPaxos) with efficient grouping algorithms. Then, for the characteristics that data is offloaded frequently, we design two synchronization strategies to ensure the consistency of the data cached in the edge-cloud and the central-cloud, respectively. Experiments show that, our proposed grouping-based consistency protocols of EECHSS can improve the availability as much as possible while ensuring data consistent.

INDEX TERMS End-edge-cloud hierarchical storage system, adaptive consistency, grouping algorithm, synchronization strategy.

I. INTRODUCTION

As the explosive growth of big data, cloud storage system has been deployed widely in the past few years due to its satisfactory advantages, such as convenient data management, fast access and reliable read [1]. However, with the development of the Internet of Things (IoT) technology [2], [3] and the fifth generation(5G) mobile communication [4], Internet connection devices increase rapidly. According to the Cisco Mobile Visual Networking Index (VNI) report, global IP data traffic will reach 3.3 ZB by 2021, and the amount of mobile data generated by terminal devices will occupy 63% of global data traffic, which is 6.7 times higher than that of 2016. By 2022, the number of devices connected to the Internet will reach 28.5 billion units, which will produce massive amounts of data every day. The high-speed growth brings a serious challenges to the capacity, the performance and the power consumption of cloud storage system. In addition, with the extensive use of deep learning [5] and augmented reality (AR) technologies [6], i.e., automatic driving [7] and smart home [8], terminal devices needs the ability of real-time data processing. However, traditional end-cloud architecture adopts centralized management, in which all data flows to the central-cloud (CC), which causes great overhead

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

of network bandwidth and makes congestion in CC [9]. Moreover, CC may require data transmitted far across geographical distances with great delay, which is difficult to meet the real-time requirements.

To solve the above problem, edge-cloud (EC) [10]–[14], a new type of cloud storage architecture is deployed near the terminal data sources. The EC storage consists of the devices with computing and caching abilities, e.g., base stations, switches and routers [15] [16]. Divided independently due to geographic areas, ECs have the responsibilities for processing data generated in neighboring terminal devices, which greatly shortens the physical distances of data generation, data calculation and data storage, and provides ultra-reliability and low-latency access services [17]. The end-edge-cloud hierarchical storage system (EECHSS) [18], which aims to sink some computing and storage tasks from CC to ECs. It is appropriate for the user's needs by reducing service delay and communication overhead at the expense of redundantly storing data. Therefore, EECHSS has attracted much attention by academia and industry [19], [20].

Distributed storage technology is the foundation of cloud storage system [21], mainly related to storage, organization and management of massive data. Replicates among multiple servers can ensure strong fault-tolerance and high availability. However, in replication, if only a part of servers complete the copy, the client accesses the same data of different servers, which will cause data inconsistency and system unreliable. Therefore, data consistency is the most basic precondition to meet in cloud storage system, and a variety of data consistency protocols are widely studied [22]–[25].

Since end-cloud storage is recognized and used, the current data consistency protocols is mostly designed of ensuring CC's data consistency. However, there are many types of applications in the current system, but the consistency level of the protocol is fixed. Different applications have different requirements for data consistency, so a single level of consistency can no longer meet system requirements. Besides, compared with the end-cloud system, EECHSS adds the ECs with storage and computing capabilities. ECs store part of data of CC. The existing protocol can only guarantee the consistency of the data inside the EC and the CC, but cannot guarantee the consistency of the data shared by the EC and the CC, which will cause hierarchical system to be untrusted. It can be seen that the existing consistency protocols cannot solve this data consistency problem. At present, many researches are studied the data consistency between ECs and CCs [26], but there is no outstanding promotion on the data consistency research of the EECHSS.

Therefore, based on the characteristics and the requirements of EECHSS, we attempt to design a data consistency protocol according to the workflow of EECHSS. Our main idea is as follows: after the terminal transmits the request to the EC, an adaptive grouping-based consistency protocol (GM-Paxos) is designed for EC with high availability. Considering the large number of terminals and the high probability that clients send requests simultaneously, we choose the Multi-Paxos protocol with the leader [27] as the benchmark protocol to avoid high latency caused by request conflicts. In addition, in the process of passing the ECs' requests to CC, we devise a data synchronization strategy, avoiding the congestion as many requests are sent to CC simultaneously. Then, after ECs pass data to CC, another adaptive grouping-based consistency protocol (GEPaxos) is designed to meet the high consistency requirement of CC. Because the number of ECs and the possibility of request conflicts are both small, the EPaxos protocol without leader [28] is selected as the benchmark. Lastly, we design a consistency strategy to ensure the consistency of the shared data between CC and ECs in EECHSS.

The main contributions of this paper are summarized as follows:

- Since EC needs low latency and high QoS, three metrics, i.e., average transmission delay, consistency degree and relevance degree are defined. The priority of each server is divided according to the three metrics, and the best primary group consisting of servers with high priority is selected by a grouping algorithm. GM-Paxos protocol is designed to send client requests only to the primary group. After the primary group is consistent, the request is considered as the executed one, to reduce the request response delay and to improve the throughput.
- When EC sends information to CC after the request execution, in order to avoid the waste of uploading the same data, we devise a data synchronization strategy. For data that is frequently changed in the short term, the upload time interval is adaptively selected according to the frequency of access to CC, so that the EECHSS consistency can be ensured with minimized wasting resource.
- Because CC provides data for the EECHSS and each EC stores different contents, we defined two metrics, average delay and related consistency. The GEPaxos protocol is devised to remove some of the servers with lower performance, and form a primary group for each EC. The servers out of primary group require strong consistency, and servers in the primary group require lazy release consistency [29], which can ensure that CC still has better availability with high consistency.
- In order to ensure the consistency of data shared among CC and ECs, we record the storage state of each EC in the CC. When CC's data is updated, it will send the message to the ECs' data. The strategy ensures that the data in EECHSS can be updated with high efficiency and low consumption, and the data shared by CC and ECs is consistent.

The rest of the paper is organized as follows. Section II presents the system architecture of EECHSS and gives some theoretical analysis. The grouping-based protocols are proposed and devised in Section III. Section IV provides two synchronization strategies. Finally, we conclude this work and present future plans in Section V. For the explanation of the parameters in this paper, please refer to TABLE 1.

TABLE 1. Parameters summary.

parameter	description
TE	Terminal equipment
E_M	Edge-cloud M
CC	Central-cloud
S_i	Server i
N	Number of servers
t_i	Average transmission delay of server i
t_{in}	Transmission delay from server i to server n
c_i	Consistency degree of server i
$Q_{\mathrm{complete}-i}$	Number of requests that server i has processed
$Q_{\mathrm{sum}-i}$	Number of client requests received by server i
r_i	Relevance degree of server i
$V_{\mathrm{server}-i}$	Number of accesses to server i per unit time
$V_{\mathrm{sum}-i}$	Total number of accesses to server i
e_i	efficiency indicator of Server i
$\boldsymbol{e}$	Vector of each server efficiency indicator, $\boldsymbol{e} = [e_1, e_2, \cdots, e_N]$
j_i	The state of server i, $j_i \in (0, 1)$, takes 1 in the primary group, otherwise takes 0.
$\boldsymbol{j}$	A vector reflecting whether each server belongs to the primary group, $j = [j_1, j_2, \cdots, j_N]$
$\boldsymbol{j}_j$	The j-th possible combination of $\boldsymbol{j}$
$\boldsymbol{j}_{\boldsymbol{j}_i}$	The state of server i in $\boldsymbol{j}$, $j_i \in (0, 1)$
P_C	The ratio of the number of synchronization servers to the total number of servers that a request has completed
E_C	Half of the sum of all server efficiency indicators
F	Efficiency function, $F = \boldsymbol{e} \cdot \boldsymbol{j} - P_C$
$F_{\boldsymbol{j}_i}$	Efficiency function in the j combination of $\boldsymbol{j}$
G_p	The primary group
G_S	The collection of servers other than the primary group
t_{ia}	Transmission delay of server i to agent server
rc_i	Relevant degree of consistency of server i
$Q_{(a-complete)-i}$	The number of requests which the agent has completed completed by server i
$Q_{(a-sum)-i}$	The number of requests the agent has completed
d_i	agent elimination indicator of server i
$\boldsymbol{d}$	Vector composed of server efficiency indicators d_i, $d = [d_1, d_2, \cdots, d_N]$
D_C	Half of the sum of all server elimination indicators
B	Elimination function, $B = \boldsymbol{d} \cdot \boldsymbol{j} - P_C$
o	The degree of expiration of CC data
T_s	The maximum time interval from the last upload of the request to CC
λT_s	Average arrival time for requests for data in CC
t_s	Maximum time difference allowed for concurrent processing of two related requests

II. SYSTEM MODEL

The EECHSS architecture is shown in Figure 1. The system is composed of servers, data centers and devices. The device that the user can directly interact is called terminal. After receiving the user request, the terminal sends the request to the remote large server, which constitutes a CC. Installations with computing and storage abilities between the terminal and CC, i.e., base stations, switches and routers, are divided into multiple ECs according to geographic areas. The terminal is to send user requests to EC, and EC buffers a part of the data, but each EC's part is different. EC is to process the requests and to pass the requests to CC using more power. CC receives and processes the requests uploaded by each EC. EC helps CC to process a part of the transaction and EC also can directly interact with the terminal devices.

A. PROTOCOL CHARACTERISTICS

As an auxiliary device for CC, EC alleviates the problems of insufficient bandwidth, limited processing capacity, and poor experience of users. Therefore, reducing the request response time and improving the users' experience are the key requirements. When designing a protocol, we focus on achieving fast consistency, ensuring minimizing latency to improve the system availability. Considering the large number of terminal devices, the probability that multiple clients send requests to EC at the same time is so high. To avoid high latency caused by high-concurrency request conflicts, we choose Multi-Paxos protocol with the leader as the benchmark, and the GM-Paxos protocol is proposed firstly.

Moreover, CC is responsible for providing data to each EC. Once an error occurs, the related EC and its corresponding terminals will receive incorrect data, and the system will become unreliable. We choose to sacrifice a part of availability to achieve high consistency when designing the consistency protocol, but it is still a eventual consistency protocol, and the degree of consistency is lower than the strong consistency. When the number of ECs is small and EC sends a request to CC, the possibility of requesting collision is very small. To improve efficiency, the EPaxos protocol without leader is selected as the benchmark, and the GEPaxos protocol is further proposed.

B. PROTOCOL PROCESS

Based on the workflow of consistency protocol in EECHSS as shown in Figure 1, and the characteristics and the requirements, Figure 2 shows the detailed implementation of the two situations in which data transmission occurs. When the terminal device makes a request, the request is first sent to EC by default, and EC determines whether it has the ability to process the request depending on the request of the data cached. If EC can process, as the Situation 1 in the $\mathrm{E_A}$: EC performs the feedback to terminal based on the GM-Paxos protocol, and judges the frequency of updated data, and adaptively selects the time of data synchronized to CC. After CC completes the synchronization based on GEPaxos, the updated data is sent to the relevant EC that caches the data according to the record of the content of each EC's cache. Between the stage where the EC executes the request and the stage where the request is sent to the CC, there is a synchronization process, and the time required for this process is the gap in Figure 2. Therefore, the arrow is discontinuous. If EC cannot be processed, as the Situation 2 in the $\mathrm{E_B}$: EC acts as a relay to forward the request to CC. After CC performs the request based on the GEPaxos protocol, CC can give a feedback, and EC can give a feedback to the terminal. CC also sends the updated data to the relevant EC that caches the data. The feedback link is represented by dashed line in Figure 1.

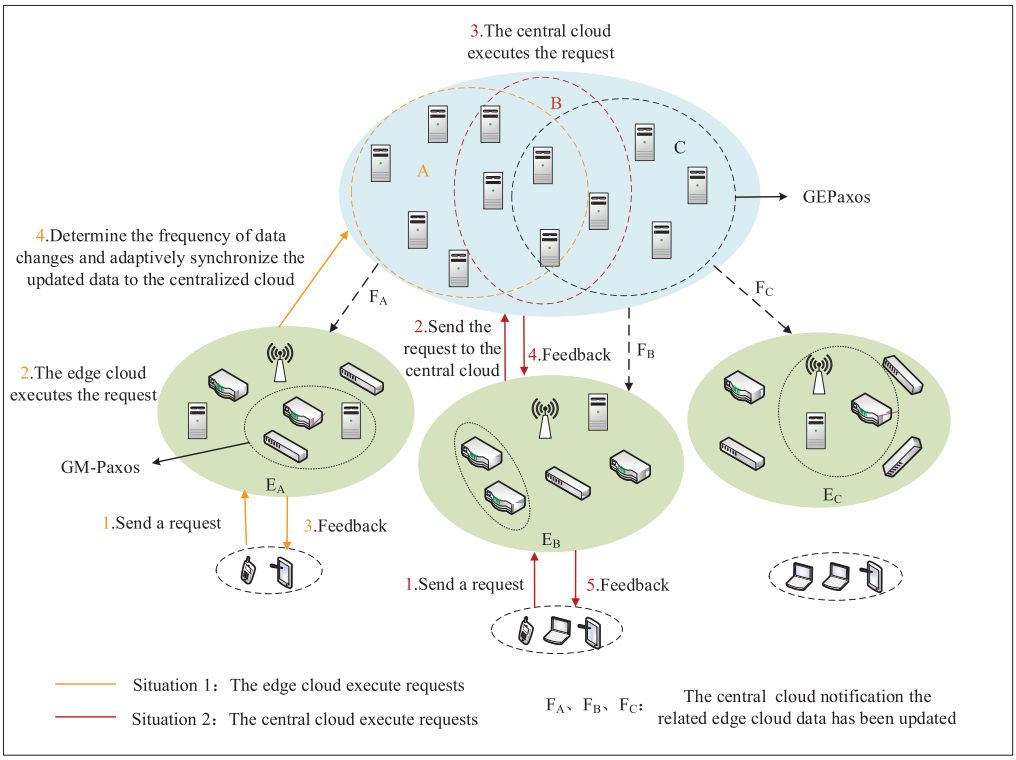
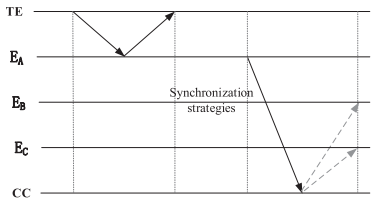
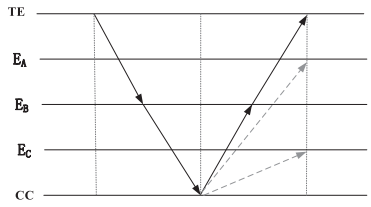


FIGURE 1. The end-edge-cloud hierarchical storage system architecture diagram and working principle.



(a) The execution process of the system protocol in situation 1



(b) The execution process of the system protocol in situation 2

FIGURE 2. The end-edge-cloud hierarchical storage system architecture execution process.

F_A, F_B and F_C represent the feedbacks to E_A, E_B and E_C, respectively.

C. DIFFICULTIES IN PROTOCOL DESIGN

In the above scenarios, the problems mainly exist in the consistency protocol procedure, which can be summarized as follows:

- Since existing protocols are more general-purpose protocols, and can not dynamically adjust the level of consistency according to actual environmental requirements [30], we need to design the protocols based on the characteristics of ECs, and requirements of CC with high availability.
- In the procedure of EC passing the request to CC, considering all requests sent directly, the congestion is easy to occur, making CC processing limited. Even if the request is transferred to CC, CC needs to be queued for the execution, which will cause large delay. After CC data is updated, the traditional method is to send the update information to all ECs by broadcasting, which does not consider whether each EC needs to know this information.

III. PROTOCOL DESIGN

A. GM-Paxos PROTOCOL

As shown in Figure 3, Edge-cloud system has N servers distributed inside each EC, and each server provides services to several clients. When a client sends a request to a server (the server is recorded as server-client), the Multi-Paxos protocol requires the server-client to send the requested content to all other servers. And the request is considered completed when more than half of the servers reply. The consistency level of the protocol is fixed, and the upper bound of bandwidth will be reached quickly, that results in degraded performance of the system. Therefore, we propose GM-Paxos to solve this problem.

We consider the case where the request content is first sent to a partial server. Once the partial servers complete

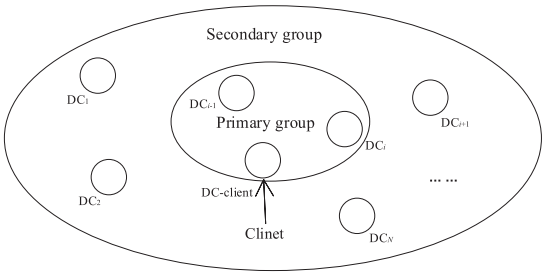


FIGURE 3. Edge-cloud system framework.

the synchronization, the request execution is considered as completed. When the bandwidth resources are sufficient, the server-client sends the requested content to other servers. We collect these servers that first receive the request into the primary group, while the other servers are classified as secondary group. In this way, the system can complete the request with a smaller amount of transmitted data in a shorter time, which will increase the processing power of the system. The number of primary group determines the level of consistency of the request.

Since the servers are allocated in various environments, the processing abilities to requests and the qualities of communication are different. Therefore, we can choose the primary group by the difference of servers. In data replication, latency is directly affecting the users' experience. The servers have different communication qualities, so the average transmission delay will vary. We can choose the average transmission delay as an indicator. Processing ability of one server is also important, because the more unprocessed requests, the longer the queue time and the lower the consistency of the server. Therefore, we definite the consistency degree as the second metric. Moreover, servers have different sensitivity to the request contents. Some servers have high requirements on the real-time performance of the data, and the amount of access is large. Therefore, we use the relevance degree as the third metric. The three metrics for selecting the primary grouping are defined as follows.

Definition 1 Average Transmission Delay: *Assume that the transmission delays from a single server S_i to other servers are denoted by $t_{i1}, t_{i2}, \cdots, t_{i(N-1)}$ and the average transmission delay of a single server denoted by t_i is defined as the ratio of the total transmission delay of the server to all other servers to the total number of servers N.*

$$t_i = \frac{\sum_{i=1}^{N-1} t_{in}}{N} \tag{1}$$

When the delay is small, the system has short response time and high availability.

Definition 2 Consistency Degree: *The consistency degree of a single server S_i denoted by c_i, is the ratio of the number of processed requests $Q_{complete-i}$ to the total number of client requests Q_{sum-i}.*

$$c_i = \frac{Q_{complete-i}}{Q_{sum-i}} \tag{2}$$

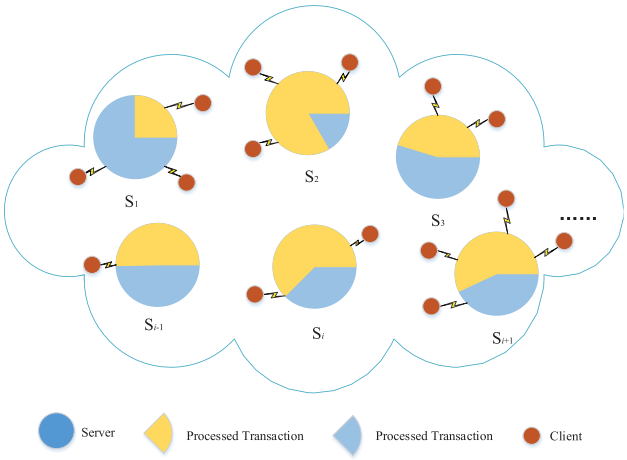


FIGURE 4. Part of edge-cloud system internal structure.

The high consistency degree of the server indicates that the server has fewer requests without synchronization, and new requests can be executed faster.

Definition 3 Relevance Degree: *The relevance degree of a single server denoted by r_i, is the ratio of the number of visits to the server V_{DB-i} to the total number of visits V_{sum-i} in a period of time.*

$$r_i = \frac{V_{DB-i}}{V_{sum-i}} \tag{3}$$

If the server with a large number of visits is synchronized first, the probability of the client accessing the latest data will be increased, and the system reliability will be guaranteed.

The internal structure of the system is shown in Figure 4. Different servers have different characteristics. In terms of the consistency degree c_i, the yellow parts represent the processed transaction and the blue parts represent the unprocessed transaction. The greater the proportion of the yellow parts, results in the higher the consistency degree. The relevance degree r_i represents the number of clients connected to the server. If the number of clients is large, the relevance degree will be high. The average transmission delay t_i denotes the distance between two servers, and if one server is far away from most servers, delay will be large. Based on these three metrics, we can select several servers with high consistency degree, high relevance degree, and low average transmission delay as the primary group, which will improve the consistency level and availability. It should be noted that, the primary group must be able to represent the majority to ensure the uniqueness of the decision. The system flowchart of GM-Paxos is shown in Figure 5, and seven server named A to G are allocated in the system.

STEP 1: Client sends a request to the nearest server G.

STEP 2: G sends a request to collector D to get the table data, where the collector is randomly selected. Each server regularly sends the number of requests completed, and the number of visits and the average transmission delay per unit time to the collector.

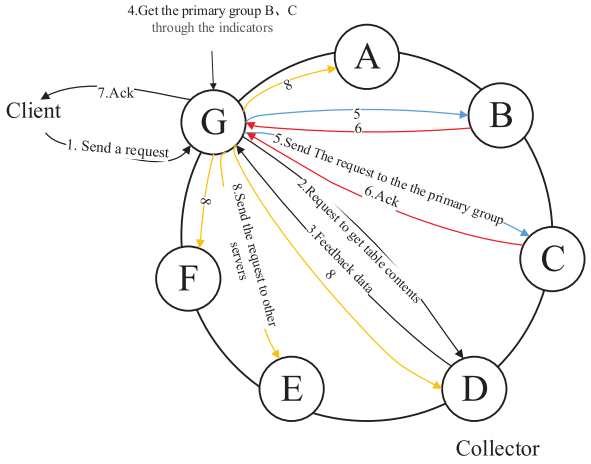


FIGURE 5. System flowchart of GM-Paxos.

STEP 3-4: G obtains the latest metric values from D and get the primary group which is composed of B and C according to the metrics.

STEP 5-7: G sends the request to the servers in the primary group. And after receiving the feedback information from B and C, it reports back to the client that the request has been completed.

STEP 8: G sends the request to other servers during the bandwidth free period and the system will achieve eventual consistency.

Each server has the above three metrics, and we combine these three metrics with a formula to get a efficiency metric defined as follow.

Definition 4 Efficiency Metric e_i: Efficiency metric e of a single server is the ratio of the relevance degree of the server and the average transmission delay to the waiting delay.

$$e_i = \frac{r_i}{t_i + \beta(1 - c_i)}, \tag{4}$$

where β is the system parameter. The efficiency metric e_i is used to judge the priorities of the servers to form the primary group.

We define that the sum of efficiency metrics of the servers in the primary group must be greater than half of the sum of all servers, to ensure that there are no two primary groups at the same time. And we also claim that the number of primary group does not exceed half of the number of all servers, which can reduce the response time of the request.

Under the above requirements, we can get multiple eligible sets, so we need to construct an optimization objective function. By weighing the efficiency metric and the number of primary groups, we attempt to find the best primary group that meets the high efficiency metric and the number of servers within primary group is small.

$$\begin{aligned} J_0: \ & \max_{\boldsymbol{j}} \ (\boldsymbol{e} \cdot \boldsymbol{j} - P_C) \\ & \text{s.t. } \boldsymbol{e} \cdot \boldsymbol{j} \geq E_C \\ & \sum_{i=1}^{N} j_i \leq \frac{N}{2}, \end{aligned} \tag{5}$$

where $\boldsymbol{e}$ is the vector composed of server efficiency metrics, which can be expressed as $\boldsymbol{e} = [e_1, e_2, \cdots, e_N]$. $\boldsymbol{j}$ is the vector reflecting whether the server belongs to the primary group, which can be expressed as $\boldsymbol{j} = [j_1, j_2, \cdots, j_N]$ where $j_i \in [0, 1], i = 0, 1, \cdots, N$. "1" indicates that the server is in the primary group, and "0" indicates that the server is outside the primary group. P_C is the percentage of the servers whose requests have been synchronized, expressed as:

$$P_C = \alpha \frac{\sum_{i=1}^{N} j_i}{N}, \tag{6}$$

where α is the system parameter. E_C is a threshold that represents half of the total server efficiency metric. The selected primary group must meet this threshold to represent the majority. E_C specific manifestation is:

$$E_C = \frac{1}{2} \sum_{i=1}^{N} e_i. \tag{7}$$

The optimal group selection algorithm is as following, and $\boldsymbol{e} \cdot \boldsymbol{j} - P_C$ is defined as the efficiency function F.

G_p is the primary group and G_s is the secondary group. When a client sends a request to the system, the efficiency index e_i of each server in the system is first obtained, and then the vector $\boldsymbol{e}$ and the threshold E_C are obtained (steps 1-4). By traversing all possible forms of $\boldsymbol{j}$, we find a $\boldsymbol{j}$ vector that satisfies the highest efficiency function F on behalf of the majority (steps 5-12). Then we can know which servers are composed of primary group. After selecting the primary group, GM-Paxos can begin data synchronization. Compared to the Multi-Paxos, we only synchronize the servers within the primary group to reduce the response delay. Moveover, after the request is completed, the amount of data that needs to

Algorithm 1 Optimal Primary Group Selection Algorithm

Input:
The average transmission delay $t_i (i = 1, 2, \ldots, N)$
The consistency degree $c_i (i = 1, 2, \ldots, N)$
The relevance degree $r_i (i = 1, 2, \ldots, N)$
Output:
The optimal vector Opt.$\boldsymbol{j}$

```
for each i ∈ G_p ∪ G_s do
    Compute e_i
end for
Compute e
Compute E_C
for each j_j do
    if e · j_j ≥ E_C && Σ_{i=1}^{N} j_{j_i} ≤ N/2 then
        Compute F_j
        if F_j ≥ MaxF then
            MaxF = F_j
            Opt.j = j_j
        end if
    end if
end for
```

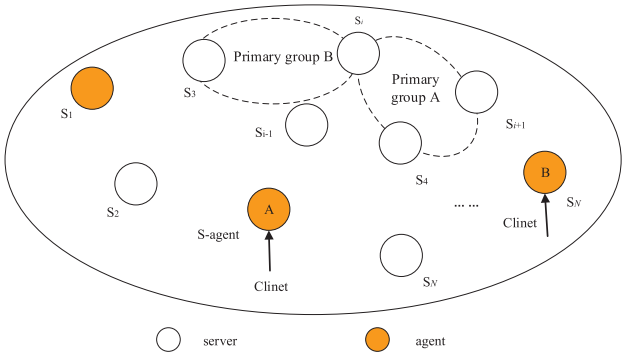


FIGURE 6. Central-cloud system framework.

be transferred will decrease and the throughput of the system will increase.

B. GEPaxos PROTOCOL

CC is a distributed storage system consisting of a set of high-capacity, high-computing servers. It requires a data consistency protocol to address data consistency issues across servers within the system. Since the EECHSS uses the data of CC as the standard, if the consistency level of CC is low, EC may access the expired data, and the data accessed by the terminal is expired data, which causes the system to be unreliable. Because the data cached in EC differs and is located differently, the optimal server for selecting interactions is different for each EC. In order to ensure that each EC upload request can be quickly responded, we select an agent in CC for each EC. If three edge-clouds are assumed, there are three agents here. Requests from EC are passed directly to their respective agents, which are scheduled to execute. In addition, we select a specific primary group for each EC based on the agent, the average transmission delay of each server in CC, and the degree of consistency. While ensuring high system consistency, the protocol ensures that EC request is executed as quickly as possible. CC system framework is shown in Figure 6.

The specific execution process is divided into two parts. When any EC transmits data to CC for the first time, CC first obtains the agent of EC through election, and then obtains the primary group by combining the metric and the group algorithm. In the case of agent mode, EC will pass the request directly to the agent, and the agent interacts with the primary group to schedule the execution request. When the agent fails, the agent is re-selected before continuing to transfer data.

The method of electing an agent is to find a server that completes the latest request for EC and has the lowest average latency, because such a server is very computationally capable and able to communicate quickly with others. The specific election process is shown in Figure 7. Servers have three states: looking, following and leading. The voting information consists of election round, request numbers, and average delays. Among them, the election number is automatically added after each vote, and the request number is also monotonically increasing. The voting rule is to compare the

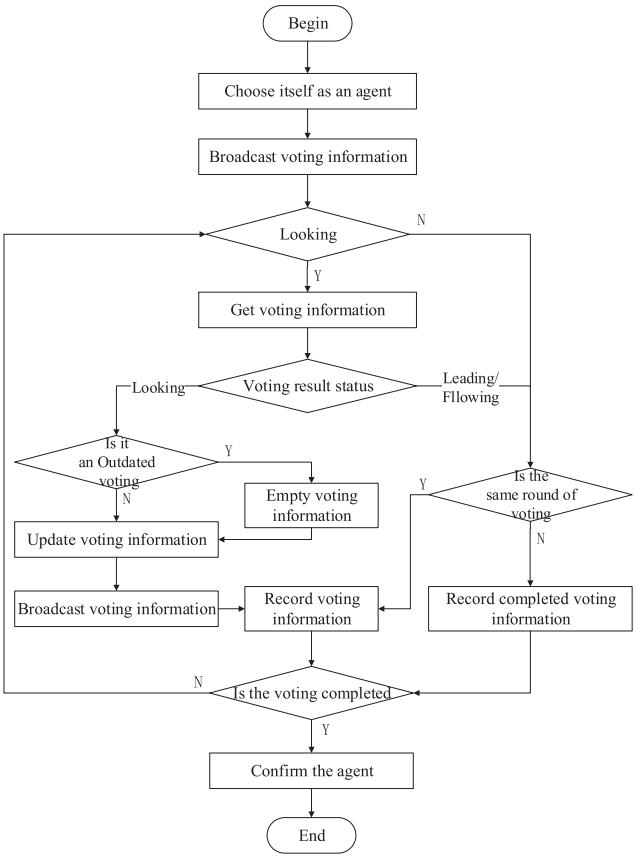


FIGURE 7. Edge-cloud system flow chart.

rounds first. If the externally received information round is greater than the internal round, the voting needs to be changed to the received external voting, otherwise the voting will not be changed. If the rounds are the same, the request numbers of the external information and the internal information are compared. If the external information is large, the server need to change the voting, otherwise the voting will not be changed. If the round and the request number are the same, the delay is compared. If the delay of the external information is small, the voting will be changed, otherwise it will not change. In this way, we can choose an agent for each EC.

After finding the corresponding agent for each EC, we will group each EC based on the agent. Because EC request is passed directly to the agent, it is sent by the agent to each server for execution. Therefore, the agent has different transmission delays from other server locations, and the transmission delay is an important factor affecting the response time of the request. The average transmission delay is used as an metric to judge the performance of the server. In addition, since there are multiple agents, the order of requests received by each server may be different, so the server does not have the same degree of completion of cache data related requests in each EC. The higher the completion of the request, the shorter the queuing time and the faster execution of the request, so the relevant completion will be used as another metric to

judge server performance. The two metrics are extended by Definitions 1 and 2 respectively.

For GEPaxos, the transmission delay of a single server S_i is only related to the corresponding agent. So according to Definition 1, the average transmission delay t_i is

$$t_i = t_{ia}, \tag{8}$$

where t_{ia} is the transmission delay of the server to the agent. When the delay is small, the system has short response time and high availability.

For GEPaxos, The degree of relevant consistency is the percentage of requests that the server has completed in the request that the agent has completed. So according to Definition 2, the average transmission delay rc_i is

$$cr_i = \frac{Q_{(a-complete)-i}}{Q_{(a-sum)-i}}, \tag{9}$$

where $Q_{(a-complete)-i}$is the number of requests completed by the server in the agent's completed request, and $Q_{(a-sum)-i}$ is the number of requests completed by the agent. The high relevant consistency degree of the server indicates that the server has fewer requests without synchronization and new requests can be executed faster.

Every agent will record the transmission delay with each server and update it regularly. At the same time, in CC, a contact server is randomly selected to store the cache information of each EC, and the consistency of each server is obtained. Since CC pursues high consistency, we hope to form a primary group by finding servers with high transmission delay and low correlation consistency. When an agent in CC receives the request, it preferentially sends the request to a server other than the primary group, and reduces the performance of the entire system by reducing some of the poorly performing servers in the system. In addition, because CC requires high consistency, the number of servers within primary group must be less than half.

In order to judge the performance of the server, we combine the above two metrics with a formula to obtain a new metric called the agent-elimination metric, which is defined as follows:

Definition 5 Agent-Elimination Metric d_i: Definition Efficiency metric d_i of a single server S_i is the ratio of the degree of relevant consistency and the transmission delay.

$$d_i = \frac{t_i}{\alpha(1 - rc_i)}, \tag{10}$$

where α is the system parameter. The agent-elimination metric d_i is used to judge the priorities of the servers to form the primary group.

We define that the sum of the elimination metrics for the servers in the primary grouping must be greater than half of the total phasing metrics for all servers to ensure that there are no two primary groups at the same time. We also require that the number of primary groups not exceed half of all servers, which can reduce the response time of requests and increase the level of consistency of the system after synchronization is completed. Since the elimination metrics of each server are different, we can guarantee that even if the number of primary groups is less than half of the number of all servers, the sum of the elimination metrics can still exceed half of the sum of all servers, and the uniqueness of the primary group can be guaranteed.

According to the above requirements, we can get multiple eligible collections, so we need to build an optimization objective function. By weighing the elimination metrics and the level of consistency of the system, we try to find the primary grouping that meets the high elimination metric, and the number of primary groups is as small as possible.

$$\begin{aligned} J_1: \ & \max_{\boldsymbol{j}} \ (\boldsymbol{d} \cdot \boldsymbol{j} - P_C) \\ & \text{s.t.} \ \boldsymbol{d} \cdot \boldsymbol{j} \geq D_C \\ & \sum_{i=1}^{N} j_i \leq \frac{N}{2}, \end{aligned} \tag{11}$$

where $\boldsymbol{d}$ is the vector composed of server agent-elimination metrics, which can be expressed as $\boldsymbol{d} = [d_1, d_2, \cdots, d_N]$. D_C is a threshold that represents half of the total server agent-elimination metric. The selected primary group must meet this threshold to represent the majority. E_C specific manifestation is:

$$D_C = \frac{1}{2} \sum_{i=1}^{N} d_i. \tag{12}$$

The optimal group selection algorithm is as following, and $\boldsymbol{d} \cdot \boldsymbol{j} - P_C$ is defined as the elimination function B.

When a client sends a request to the system, the agent-elimination metric D_i of each server in the system is first obtained, and then the vector $\boldsymbol{d}$ and the threshold D_C are

Algorithm 2 Optimal Elimination Primary Group Selection Algorithm

Input:
The transmission delay $t_i a(i = 1, 2, \ldots, N)$
The relevant consistency degree $rc_i a(i = 1, 2, \ldots, N)$
Output:
The optimal vector Opt.$\boldsymbol{j}$
for each $i \in G_p \cup G_s$ **do**
  Compute d_i
end for
Compute $\boldsymbol{d}$
Compute D_C
for each $\boldsymbol{j}_j$ **do**
  if $\mathbf{d} \cdot \boldsymbol{j}_j \geq D_C \&\& \sum_{i=1}^{N} \boldsymbol{j}_{j_i} \leq \frac{N}{2}$ **then**
    Compute B_j
    if $F_j \geq MaxB$ **then**
      MaxF $= B_j$
      Opt.$\boldsymbol{j} = \boldsymbol{j}_j$
    end if
  end if
end for

obtained (steps 1-4). By traversing all possible forms of $\boldsymbol{j}$, we find a $\boldsymbol{j}$ vector that satisfies the highest elimination function B on behalf of the majority (steps 5-12). Then we can know which servers are composed of primary group. Because we want to eliminate the servers in the primary group and prioritize the servers outside the primary group, the set of servers that are preferentially synchronized is $\mathbf{I} - \boldsymbol{j}$, where $\mathbf{I}_{1\times N} = [1, 1, \cdots, 1]$.

Select agents separately in CC for each EC. The server's elimination priority is determined according to two metrics: the average transmission delay and the relevant consistency level. By selecting the best-eliminated primary group to circumvent the request response time of the entire system due to a few poorly performing servers. When EC transmits a request to CC, the server outside the primary group is preferentially synchronized. When the server outside the primary group feeds back the confirmation message, it considers that the execution of the request is completed, and ensures the high availability of the data in CC while improving the availability of the system as much as possible.

IV. DATA SYNCHRONIZATION STRATEGY

The biggest difference between the EECHSS and traditional end-cloud system is the addition of ECs, which complicates the form of data transmission. We need to pay attention to the data interaction efficiency between each EC and CC and the consistency of the shared data among ECs and CC.

A. DATA SYNCHRONIZATION STRATEGY FROM ECS TO CC

In the stage where EC sends the processed data or the unexecuted request to CC, EC directly transmits the data completed each time to CC. If multiple requests processed continuously are changes to the same data, it will result in a lot of bandwidth and a waste of CC computing. In addition, there may be cases where requests from each EC are queued for execution due to too many requests, resulting in high latency of the entire system. So we will judge each data that will be uploaded. If there is no other operation of the data in a short time, the data is uploaded to CC. If the data is continuously changed in a short time, only the latest data is transmitted to CC.

The specific algorithm flow of the data synchronization strategy from EC to CC is shown in Figure 8. When EC receives the request from the terminal device, it first determines whether the request can be processed. If the data is not cached by itself, the request is sent directly to CC. If the data is cached by itself, the request is updated to update the data, and it is determined whether the data has been modified in the t_s. If the data has not been modified, the data is directly synchronized to CC. If the data has been modified, it is determined whether the time from the last upload to CC exceeds T_s. When the T_s is exceeded, the data is uploaded to CC. Otherwise, it is cached in EC, waiting for a new update

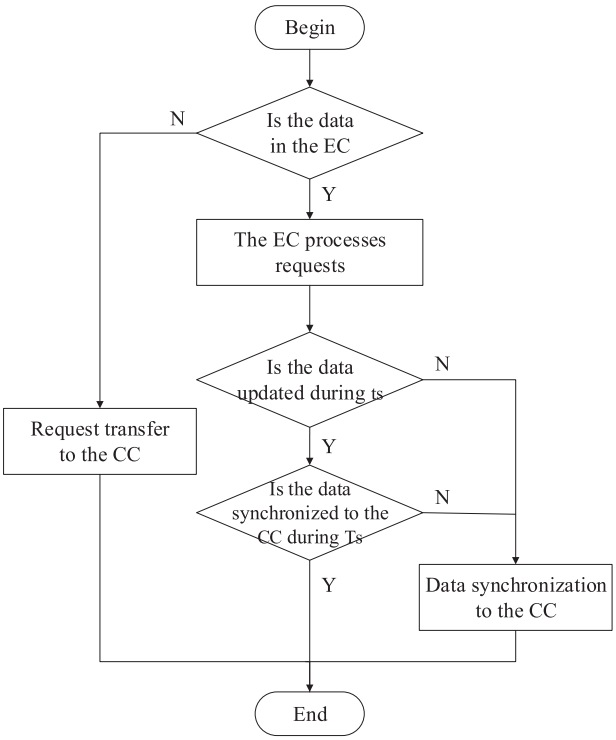


FIGURE 8. Flow chart of data synchronization strategy from a EC to CC.

operation or reaching CC after reaching the T_s.

$$P(o \le d) = \sum_{n=0}^{n=d} \frac{(\lambda T_s)^n e^{-\lambda T_s}}{n!}, \tag{13}$$

where o is the data expiration rate of CC, and d d indicates the number of times the data was requested in CC. T_s represents the average arrival rate of the request for the data in CC in the T_s time. Combined with CC feature, we take the value of d to zero, and try to ensure that CC is accessed with the latest correct data. At the same time, in order to ensure that consecutive requests for the same data in the t_s time can be processed together, it is necessary to satisfy: $t_s \le 1/2T_s$.

B. CONSISTENCY STRATEGY FOR DATA CACHED BY ECS AND CC

After CC data is updated, in order to ensure the data consistency, the update information of the data needs to be notified to each EC that caches the data. The traditional way is mainly divided into two types. One is to broadcast the update information to all ECs after each data update. Regardless of whether EC needs to be notified, the data consistency can be better ensured. But the way of broadcasting causes a large waste of resources, and the other is that EC periodically synchronizes to CC. The timing of the regular update of this method is difficult to select and the data consistency of the system cannot be guaranteed. Therefore, our approach is to concentrate the cloud storage cache information of each EC. When the cloud data is updated, the update information is only sent to EC in which the data is cached. Since the cached

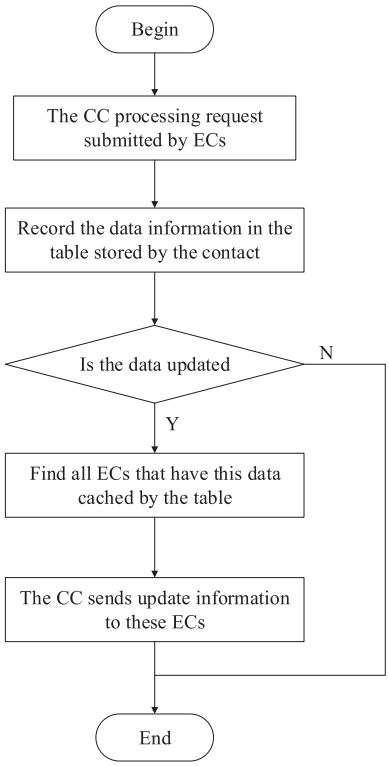


FIGURE 9. Flow chart of consistency strategy for data cached by ECs and CC.

TABLE 2. Simulations parameters.

parameter	description	value
B	Bandwidth	1Mbps
T_{E}	Average transmission delay in E_M	2-30ms
T_{CC}	Average transmission delay in CC	2-50ms
N_{E}	Number of servers in E_M	7
N_{E}	Number of servers in CC	9
m	Number of clients	0-2000
S_{req}	Size of request message	100Byte
T_{client}	Transmission delay of client	1ms
T_{ECC}	Transmission delay of E_M and CC	15ms

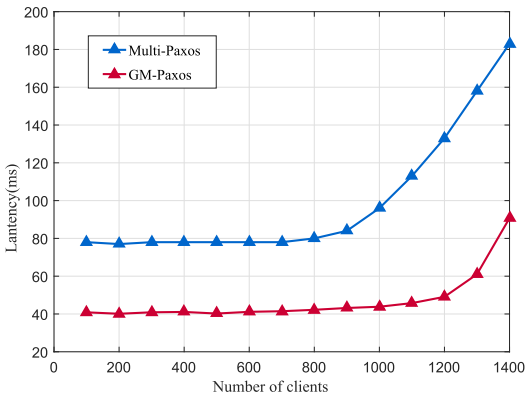


FIGURE 10. Average latency comparison between GM-Paxos and Multi-Paxos.

data of each EC is carried when submitting the request to CC, it does not cost extra resources, so it is more resource-saving than the existing solution.

The specific process is shown in Figure 9. A contact is randomly selected in CC to store cache information of each EC, and the cached information is stored in the form of a table. If the cache is full, the old cache with the smallest time-stamp is removed. When CC receives a request, it first records the content contained in the request into the table, and then determines whether the request has a data update. If the data is not updated, no further operations will be continued. And if the data is updated, all ECs that have this data cached by the table are found in the table, and the updated data is sent to these ECs.

V. SIMULATION RESULTS

We divides the simulations into three parts: the GM-Paxos protocol in ECs, the GEPaxos protocol in CC, and the data consistency protocol in EECHSS. And the parameters used are shown in Table 2.

We set 1 CC and 3 ECs in system. CC contains 9 service nodes, and each EC contains 7 service nodes. Since the client is sent near the request, the transmission delay of the client to EC node is fixed to 1 ms. EC is close to the client and is relatively far away from CC, so the transmission time from EC to CC without congestion is delayed to 15ms. The transmission delay between EC and the nodes in CC obeys a Gaussian distribution with a mean of 25 and variances of 10 to simulate the actual communication environment. The geographic area of each EC is relatively small, so the transmission delay interval between nodes is set to 2-30 ms. The value of the transmission delay value of each node in CC is set to 2-50 ms. The client is arbitrarily distributed on each node, and the client does not send the next request until the current request is completed.

A. GM-Paxos PROTOCOL PERFORMANCE IN ECS

This section evaluates the average latency and throughput of GM-Paxos through the experiments and compares them with the Multi-Paxos protocol in the same environment.

It can be seen in Figure 10 that the average delay of GM-Paxos is significantly lower than the Multi-Paxos protocol. Since the primary group is selected by three metrics: the consistency degree, the relevance degree and the average transmission delay, the number of servers in the primary group is less than half of all servers. When the number of requests is small, the bandwidth resources are sufficient. At this point, all requests are considered to be sent at the same time. However, consistency degree and average transmission delay are related to the total delay and the servers in the primary group usually have lower latency. When the server with the highest delay in the primary group completes the synchronization, time is still shorter than Multi-Paxos. When the number of clients is close to 1000, Multi-Paxos has a shortage of bandwidth resources because the request content is sent to all servers at the same time. It can be seen from the Figure 4 that the delay of Multi-Paxos is greatly increased,

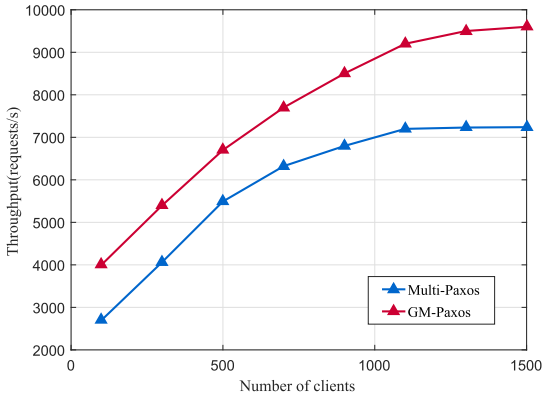


FIGURE 11. Throughput comparison between GM-Paxos and Multi-Paxos.

and a large number of requests cannot be completed quickly. Our strategy choose only sending requests to the primary group, so we use a smaller bandwidth for each request than Multi-Paxos. And when the number of clients is nearly 1200, we still keep the low latency. Meanwhile, we synchronize the servers with high relevance degree at first to ensure that servers with high data real-time requirements can access the latest data. In short, the average latency of GM-Paxos is lower than that of Multi-Paxos.

In Figure 11, we compares the throughput of our scheme and Multi-Paxos. In the figure, both curves are steadily rising in the previous period, but the throughput of our strategy is higher than Multi-Paxos. This is because we only send the request content to the primary group at first instead of all servers. When the request execution is completed, the amount of data we transmit is much smaller than Multi-Paxos. Therefore, under the same bandwidth condition, GM-Paxos can completed more requests per unit time, and has a higher throughput. Due to bandwidth limited, the curve of Multi-Paxos tends to be gentle quickly, but GM-Paxos still maintains high throughout. Therefore, the throughput of GM-Paxos is significantly higher than that of Multi-Paxos.

B. GEPaxos PROTOCOL PERFORMANCE IN CC

Figure 12 shows the average delay of 3PC, GEPaxos and EPaxos. The EPaxos protocol is open to the public only when more than half of the nodes complete the consensus system, so the average latency is the lowest. The GEPaxos protocol combines the EPaxos protocol with group technology, and the decision conditions are changed from more than half of the nodes to the nodes outside the main group. When the system is open to the outside, the GEPaxos protocol needs to synchronize more nodes than the EPaxos protocol, so the average latency is also higher than the EPaxos protocol. However, because the GEPaxos protocol picks out individual nodes with poor performance through the group technology, it avoids the delay caused by waiting for these nodes to execute requests. In fact, in the case of a node with poor performance, the response time of the strong agreement protocol is almost half of the time waiting for the node with poor performance to receive and execute the request.

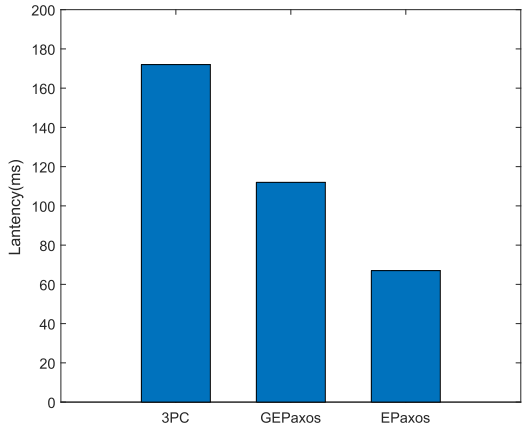


FIGURE 12. Average delay comparison among 3PC, GEPaxos and EPaxos.

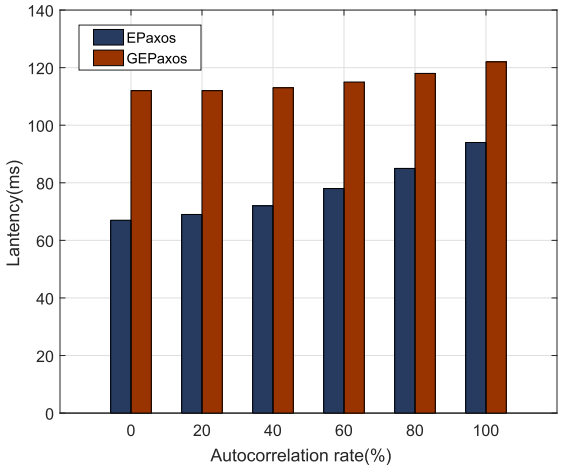


FIGURE 13. Average delay comparison between GEPaxos and EPaxos with the autocorrelation rate.

Therefore, compared with 3PC, the average delay of GEPaxos is significantly reduced.

Figure 13 shows the effect of autocorrelation rate on GEPaxos and EPaxos. The autocorrelation rate [31] is the probability of a specific causal relationship between two commands, that is, the probability that two commands are conflicts. A high autocorrelation rate means that more commands need to maintain a sequential relationship and cannot be executed concurrently. Because the EPaxos protocol allows unrelated requests to be executed in parallel, if the concurrent requests are related and then degenerate into Basic Paxos, the overall latency is greatly increased. Therefore, as the autocorrelation rate increases, the delay of the EPaxos protocol becomes larger and larger. Since the GEPaxos protocol establishes agents, only the agents have the right to execute the request. Requests can be executed in parallel between the agents, but the requests executed by each agent execution are ordered and the agent corresponding to each client is relatively stable. Therefore, the probability of a conflict between requests is low. Compared with the EPaxos protocol, the GEPaxos protocol is less affected by the correlation rate, and as the autocorrelation rate increases, the delay between

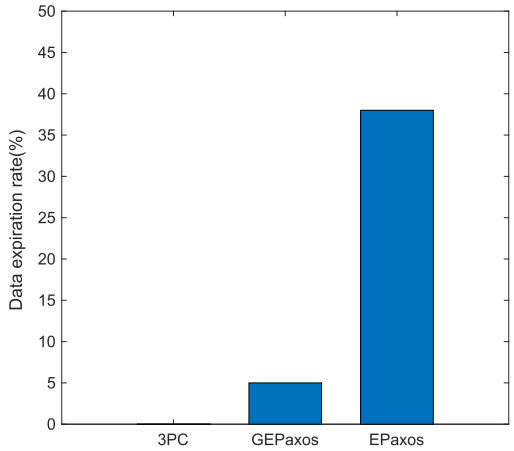


FIGURE 14. Data expiration rates of 3PC, GEPaxos and EPaxos.

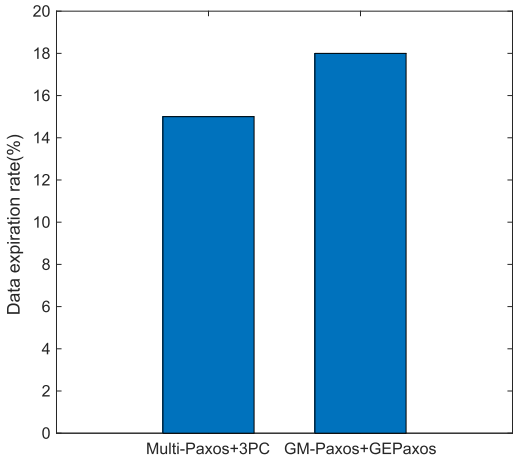


FIGURE 15. Data expiration rate of Multi-Paxos+ 3PC and GM-Paxos + GEPaxos.

the GEPaxos protocol and the EPaxos protocol becomes smaller and smaller.

Users may be able to tolerate inconsistent data acquisition when acquiring data, but only if the degree of inconsistency is within acceptable limits. This paper uses the expiration of system data as an metric to determine the level of consistency. The expiration degree refers to the probability that the customer accesses the expired error data when accessing a certain data to the system. By comparing the degree of data expiration, the level of consistency for the user can be obtained. In general, the higher the level of consistency, the more reliable the system.

Figure 14 shows the data expiration of the strong consistent protocol 3PC, GEPaxos protocol and EPaxos protocol. Because strong consistency requires all nodes to complete the request system is open to the outside, there is no possibility of accessing expired erroneous data, so the data has an expiration of 0. The EPaxos protocol is the classic Paxos class final consistency protocol. When more than half of the nodes are consistent, the system is open to the public. At this point, only 51 % of the nodes in the system are consistent, and the user has a certain probability of accessing the expired data. The GEPaxos protocol uses group technology to change the decision condition from more than half to exclude nodes in the primary group. In the case of a node with poor performance, more than half of the nodes outside the primary group complete the request and consider the request to be completed, and the conditions are more strict. Nodes that have been synchronized when the request completes will be greater than 51 % but less than 100 %. Therefore, the GEPaxos protocol will likely be accessed by users to expired data. However, since the selection of the group is related to the amount of access by the node, the node with a large amount of user access will not appear in the primary group. Therefore, the probability of accessing expired data by the GEPaxos protocol is low, and the level of user consistency is high.

As shown in Figure 12 and Figure 14, the GEPaxos protocol has a similar level of user consistency compared to 3PC, but the average latency is much lower; Compared with the EPaxos, the user consistency level of GEPaxos is greatly improved, although the average delay is higher than the EPaxos. However, it can be seen from Figure 13 that as the autocorrelation rate increases, the gap between the average delays of the two protocols becomes smaller.

C. DATA CONSISTENCY PROTOCOL PERFORMANCE IN EECHSS

Although the Multi-Paxos, GM-Paxos, 3PC, EPaxos and GEPaxos have been compared, the effect of combining the two protocols in EECHSS is unknown. In addition, the effect of the synchronization strategy also needs proved. Therefore, we will analyze and compare the two protocol combinations: Edge-Cloud Multi-Paxos + Central-Cloud 3PC and Edge-Cloud GM-Paxos + Central-Cloud GEPaxos. Compare the level of consistency and latency of the systems in the two collocation schemes by simulating the analysis under the same conditions. To ensure the reliability of the system, the synchronization strategy has been added in the scheme. We compare the performance by simulating the the degree of consistency and average latency.

The data consistency level of the EECHSS is determined by the consistency of the internal data of EC and CC and the consistency of the data shared by EC and CC, not just the consistency of EC or CC. This section still uses the expiration of client access data to reflect the consistency of the system. The lower the expiration, the higher the level of consistency. The level of consistency here refers to the level of user consistency.

Since the change in the number of clients does not have a large impact on the expiration of the system data, Figure 15 directly shows the comparison of the data expiration degree of the two combination schemes when the client is 1000. As can be seen from the Figure 15, the combination scheme of EC Multi-Paxos + CC 3PC can achieve the final consistency of external services due to the strong consistency of the 3PC protocol and the completion of more than half of the Multi-Paxos. With the cooperation of the

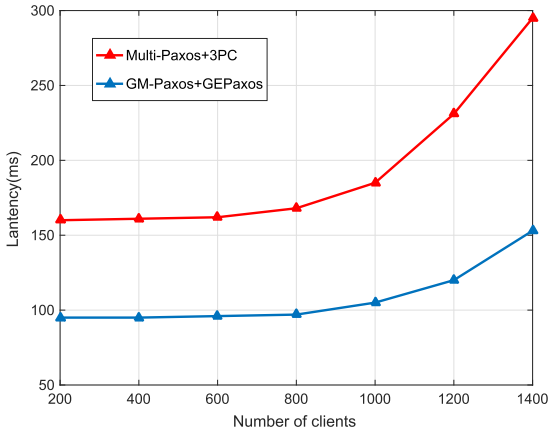


FIGURE 16. Average delay of Multi-Paxos+ 3PC and GM-Paxos + GEPaxos.

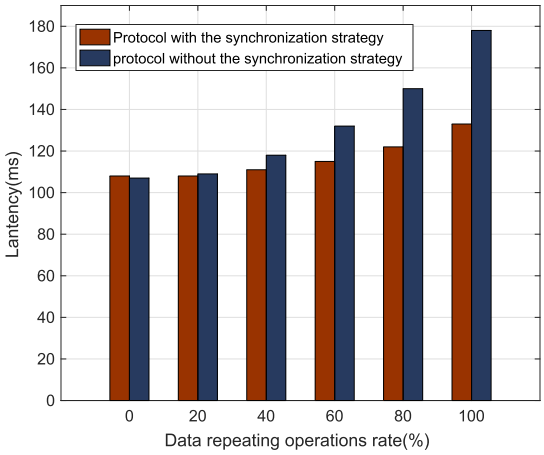


FIGURE 17. Average latency for protocols with and without synchronization policies with repeated data rates.

synchronization strategy, the average data expiration of the system is 15%. In EC GM-Paxos + CC GEPaxos designed, since the GEPaxos is not strongly consistent and the number of nodes in the main group of the GM-Paxos is less than half, the overall consistency level is relatively lower than the former, but small differences.

Figure 16 shows the delay comparison of the two combinations. As the consistency and availability of the CAP theorem are mutually balanced, the combined solution delay of EC GM-Paxos protocol + CC GEPaxos protocol is significantly lower than that of EC Multi-Paxos protocol + CC 3PC protocol. As the number of customers increases, the gap is getting bigger. Because 3PC is strong consistency, the condition for request execution is that all nodes are synchronized, and the request response time of 3PC is longer. In addition, the GM-Paxos protocol and the GEPaxos protocol proposed all prioritize the nodes with better communication quality according to the group technology, and the response time of the request is greatly shortened. When the number of clients reaches 1000, the Multi-Paxos protocol faces bandwidth limitation due to the need to send requests to all other nodes, but the GM-Paxos protocol only preferentially sends requests to nodes in the primary group, so it gets more available bandwidth. Therefore, as the number of clients increases, the delay between the two combined solutions becomes larger and larger, and the advantages of the combined solution of EC GM-Paxos protocol + CC GEPaxos protocol become more and more obvious.

For the synchronization strategy, since the consistency of the data shared between EC and CC must be guaranteed, we will focus on verifying the performance improvement of the data synchronization strategy from EC to CC. Based on EC GM-Paxos protocol, CC GEPaxos protocol, and the synchronization strategy of EC and CC shared data, we compare the delay of the system when adding the EC synchronization to CC data synchronization strategy and not adding the synchronization strategy.

Before analyzing the simulation, the concept of repeated operation probability is introduced. The probability of repeated operation refers to the probability of continuously performing multiple operations on a certain data, that is, the probability that multiple concurrent or continuous requests received by the service node are updated for the same data.

Figure 17 shows the system delay comparison of the cases of EC and CC with and without data synchronization strategy. When the probability of repeated operation is 0, the delay of the system after joining the synchronization strategy is slightly higher than that of the original system without the synchronization strategy, because the judgment process of the entire synchronization strategy takes a little time, but the case that the entire system does not have repeated operations is rare. As the probability of repeated operations increases, the system delay that does not join the synchronization policy rises rapidly, which is significantly higher than the system that joins the synchronization strategy. This is because the data synchronization strategy of EC to CC will repeat the same data in a short time. The request for the operation is first integrated in EC, and the latest request result is uploaded to CC. The above strategy avoids the queue blocking problem caused by excessive request uploading to CC, and reduces the collision probability of concurrent requests sent by each EC to CC.

VI. CONCLUSION

In this paper, we propose two grouping-based consistency protocols, named GM-Paxos and GEPaxos, in the end-edge-cloud hierarchical storage system. We combine the grouping algorithm with the existing consistency protocols, and elaborate the protocol design procedures. And we employ GM-Paxos to edge-cloud and EPaxos to central-cloud based on the system characteristics. In addition, we design the synchronization strategies to ensure the consistency of the data cached among edge-cloud and central-cloud. Experiment results prove that the proposed protocol can maximize the availability of the system while ensuring data consistent compared to traditional consistency protocols.

ACKNOWLEDGMENT

This article was presented in part at the 2019 IEEE/CIC International Conference on Communications in China (ICCC) [32]. *(Shushi Gu and Yizhen Wang contributed equally to this work.)*

REFERENCES

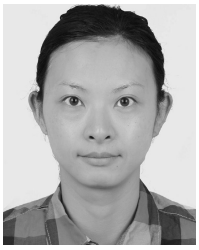
[1] P. Zhou, F. Dong, Z. Xu, J. Zhang, R. Xiong, and J. Luo, "ECStor: A flexible enterprise-oriented cloud storage system based on GlusterFS," in *Proc. Int. Conf. Adv. Cloud Big Data (CBD)*, Chengdu, China, Aug. 2016, pp. 13–18.
[2] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, "Internet of Things (IoT): Research, simulators, and testbeds," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1637–1647, Jun. 2018.
[3] Y. Liu, A. Liu, X. Liu, and M. Ma, "A trust-based active detection for cyber-physical security in industrial environments," *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6593–6603, Dec. 2019.
[4] P. Fan, J. Zhao, and C.-L. I, "5G high mobility wireless communications: Challenges and solutions," *China Commun.*, vol. 13, pp. 1–13, Dec. 2016.
[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, May 2015.
[6] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: From where we are to where we go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017.
[7] A. F. Aljulayfi and K. Djemame, "Simulation of an augmented reality application for driverless cars in an edge computing environment," in *Proc. 5th Int. Symp. Innov. Inf. Commun. Technol. (ISIICT)*, Amman, Jordan, Oct. 2018, pp. 1–8.
[8] M. Khan, B. N. Silva, and K. Han, "Internet of Things based energy aware smart home control system," *IEEE Access*, vol. 4, pp. 7556–7566, 2016.
[9] S. Yang, P. Wieder, M. Aziz, R. Yahyapour, X. Fu, and X. Chen, "Latency-sensitive data allocation and workload consolidation for cloud storage," *IEEE Access*, vol. 6, pp. 76098–76110, 2018.
[10] A. Sill, "Standards at the edge of the cloud," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 63–67, Mar./Apr. 2017.
[11] P. Skarin, W. Tarneberg, K.-E. Arzen, and M. Kihl, "Towards mission-critical control at the edge and over 5G," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, San Francisco, CA, USA, Jul. 2018, pp. 50–57.
[12] I. Lujic, V. D. Maio, and I. Brandic, "Efficient edge storage management based on near real-time forecasts," in *Proc. IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC)*, Madrid, Spain, May 2017, pp. 21–30.
[13] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.
[14] M. Linaje, J. Berrocal, and A. Galan-Benitez, "Mist and edge storage: Fair storage distribution in sensor networks," *IEEE Access*, vol. 7, pp. 123860–123876, 2019.
[15] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
[16] X. Liu, M. Zhao, A. Liu, and K. K. L. Wong, "Adjusting forwarder nodes and duty cycle using packet aggregation routing for body sensor networks," *Inf. Fusion*, vol. 53, pp. 183–195, Jan. 2020.
[17] X. Liu, T. Wang, W. Jia, A. Liu, and K. Chi, "Quick convex hull-based rendezvous planning for delay-harsh mobile data gathering in disjoint sensor networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2019.2938790.
[18] H. Liu, F. Eldarrat, H. Alqahtani, A. Reznik, X. De Foy, and Y. Zhang, "Mobile edge cloud system: Architectures, challenges, and approaches," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2495–2508, Sep. 2018.
[19] S. C. Shah, "Mobile edge cloud: Opportunities and challenges," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Las Vegas, NV, USA, Dec. 2017, pp. 1572–1577.
[20] J. Wang, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, p. 2, Feb. 2019.
[21] Q. He, Z. Li, and X. Zhang, "Study on cloud storage system based on distributed storage systems," in *Proc. Int. Conf. Comput. Inf. Sci.*, Chengdu, China, Dec. 2010, pp. 1332–1335.
[22] D. D. Akkoorath, V. Fördős, and A. Bieniusa, "Observing the consistency of distributed systems," in *Proc. 15th Int. Workshop Erlang-Erlang*, Nara, Japan, Sep. 2016, pp. 54–55.
[23] B. Calder, "Windows azure storage: A highly available cloud storage service with strong consistency," in *Proc. 23rd ACM Symp. Oper. Syst. Princ.*, Cascais, Portugal, Oct. 2011, pp. 143–157.
[24] X. Liu and P. Zhang, "Data drainage: A novel load balancing strategy for wireless sensor networks," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 125–128, Jan. 2018.
[25] W. Vogels, "Eventually consistent," *Commun. ACM*, vol. 52, no. 1, pp. 14–19, Oct. 2014.
[26] B. Pavel, "Algorithms for maintaining consistency of cached data for mobile clients in distributed file system," *Int. J. Distrib. Syst. Technol.*, vol. 8, no. 1, pp. 17–33, Jan. 2017.
[27] L. Lamport, "Paxos made simple," *ACM SIGACT News*, vol. 32, no. 4, pp. 51–58, Dec. 2016.
[28] I. Moraru, "There is more consensus in egalitarian parliaments," in *Proc. 24th ACM Symp. Oper. Syst. Princ. (SOSP)*, Farminton, PA, USA, Nov. 2013, pp. 358–372.
[29] P. Keleher, A. L. Cox, and W. Zwaenepoel, "Lazy release consistency for software distributed shared memory," in *Proc. 19th Annu. Int. Symp. Comput. Archit.*, Gold Coast, QLD, Australia, Aug. 2002, pp. 19–21.
[30] T. Jun-Feng, Z. Jia-Yao, and D. Rui-Zhong, "Date hierarchical storage strategy for data disaster recovery," *IEEE Access*, vol. 6, pp. 45606–45616, 2018.
[31] Y. Gong, C. Hu, W. Ma, and W. Wang, "CC-Paxos: Integrating consistency and reliability in wide-area storage systems," in *Proc. IEEE 22nd Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Wuhan, China, Dec. 2016, pp. 414–421.
[32] Y. Wang, Y. Wang, S. Gu, Q. Zhan, and N. Zhang, "Adaptive consistency protocol based on grouping multi-paxos," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Changchun, China, Aug. 2019, pp. 304–309.

SHUSHI GU (Member, IEEE) received the M.S. and Ph.D. degrees in information and communication engineering from the Harbin Institute of Technology, in 2012 and 2016, respectively. From 2016 to 2019, he was a Postdoctoral Research Fellow with HITSZ. From 2018 to 2019, he was a Postdoctoral Research Fellow and a Visiting Scholar with James Cook University, Cairns, Australia. He is currently an Assistant Professor with the School of Electronic and Information Engineering, Harbin Institute of Technology (Shenzhen), Shenzhen, China. He is also an Assistant Researcher with the Peng Cheng Laboratory. His current research interests include the satellite IoT, coding theory, edge caching, and distributed storage. He received the Best Paper Awards of IEEE WCSP 2015 and EAI WiSATS 2019. He also received the Outstanding Postdoctoral Award of HITSZ, in 2018.

YIZHEN WANG received the B.S. degree in communication engineering from Northeastern University, Qinhuangdao, China, in 2017. She is currently a graduate student with the Harbin Institute of Technology (Shenzhen), Shenzhen, China. Her current research interests include consistency protocols and distributed storage systems.

YE WANG (Member, IEEE) received the M.S. and Ph.D. degrees in information and communication engineering from the Harbin Institute of Technology, Shenzhen, China, in 2009 and 2013, respectively. From 2013 to 2014, he was a Postdoctoral Research Fellow with the University of Ontario Institute of Technology, Canada. He is currently an Assistant Professor with the Harbin Institute of Technology, Shenzhen, China. He is also an Assistant Researcher with the Peng Cheng Laboratory. He received the Outstanding Postdoctoral Award of HITSZ, in 2016, the Shenzhen Natural Science Award, in 2017, and the Best Paper Award of EAI WiSATS, in 2019. His current research interests include the IoT, edge computing, resource allocation, and the mobile Internet.

QINYU ZHANG (Senior Member, IEEE) received the bachelor's degree in communication engineering from the Harbin Institute of Technology (HIT), in 1994, and the Ph.D. degree in biomedical and electrical engineering from the University of Tokushima, Japan, in 2003. From 1999 to 2003, he was an Assistant Professor with the University of Tokushima. From 2003 to 2005, he was an Associate Professor with the Shenzhen Graduate School, HIT. He was the Founding Director of the Communication Engineering Research Center with the School of Electronic and Information Engineering. Since 2005, he has been a Full Professor. He has served as the Dean for the EIE School. He is also a chief of Network Communication Research Center with the Peng Cheng Laboratory. His research interests include aerospace communications and networks, wireless communications and networks, cognitive radios, signal processing, and biomedical engineering. He is on the Editorial Board of some academic journals, such as the *Journal on Communications*, *KSII Transactions on Internet and Information Systems*, and *Science China: Information Sciences*. He was the TPC Co-Chair of the IEEE/CIC ICCC'15, the Symposium Co-Chair of the IEEE VTC'16 Spring, an Associate Chair for Finance of ICMMT'12, and the Symposium Co-Chair of CHINACOM'11. He has been a TPC Member for INFOCOM, ICC, GLOBECOM, WCNC, and other flagship conferences in communications. He was the Founding Chair of the IEEE Communications Society Shenzhen Chapter. He has received the National Natural Science Foundation of China for Distinguished Young Scholars, the Young and Middle-Aged Leading Scientist of China, and the Chinese New Century Excellent Talents in University, and obtained three scientific and technological awards from governments.



XUE QIN received the B.S. degree in telecommunication engineering from the North University of China, in 2008, and the master's degree from the Conestoga College, Canada, in 2017. She is currently pursuing the graduate degree with the Department of Computing Sciences, Texas A&M University at Corpus Christi, Corpus Christi, TX, USA. Her research interests include machine learning and networking.