Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH **Open Access**

An adaptive routing strategy in P2P-based Edge Cloud

Biao Dong[1] and Jinhui Chen[2*]

Abstract

P2P-based Edge Cloud (PEC) is widely used in Internet of Things (IoT). Inevitably, the sensor data routing technology has a significant impact on the performance of PEC. Due to its prevalence and complexity, the existing routing technologies in PEC need to be optimized. Specifically, key factors such as overall network traffic, user access latency, and resource utilization of edge nodes should be considered to adapt to the dynamic requirements of user request services and network topology. In order to address the challenges produced by these factors, an adaptive routing in P2P-based Edge Cloud is proposed, which is named ARPEC. In our approach, a target edge node selection scheme based on message activity and network topology is proposed, aiming to minimize the load on edge node and user access latency. Furthermore, to minimize system overhead, sensor data routing is mapped to minimum cost maximum flow (MCMF) graph. On this basis, a target edge node selection algorithm based on a grey linear regression combination prediction model is designed, and an incremental MCMF algorithm based on belief propagation (BP) is proposed. The evaluation results show that our approach can effectively improve PEC transmission performance and user experience.

Keywords P2P, Edge Cloud, Adaptive routing, Minimum cost maximum flow, Incremental algorithm

Introduction

The vigorous development of IoT and mobile intelligent devices has brought about a trend of data sources shifting from centralized cloud data centers to distributed edge computing networks [1–3]. In IoT application scenarios based on centralized cloud computing, the data generated by sensors needs to be transmitted to cloud platform and to be processed by using the cloud computing power. Unfortunately, the massive amount of data transmission will bring a huge burden to the core network, which not only affects the normal operation of the core network, but also brings huge transmission delays and reduces user experience [4–6]. As a result, edge computing came into being, and Edge Cloud is one of the main research directions in the field. In Edge Cloud, multiple edge servers (referred to as edge nodes in the paper) are placed closer to users, and users can submits computing tasks to an edge server that is closer to them for computing. Compared with centralized cloud, Edge Cloud meets the requirements of low latency, low consumption and fast response. Therefore, the combination of Edge Cloud and IoT has become a general trend [7–10].

As is known, P2P architecture, where the neighbor nodes exchange parameters to obtain global information, is one of the mainstream architectures of Edge Cloud. Therefore, we adopt P2P architecture that has no fixed central node and can better and fully utilize the extensive distributed resources of edge nodes. Considering the important impact of sensor data routing technology on edge cloud service quality, such as reducing overall network traffic, reducing user access latency, and improving resource utilization of edge nodes, in the paper, we study an adaptive routing strategy in PEC.

*Correspondence:
Jinhui Chen
cjh@nuist.edu.cn
[1] School of Computer and Software, Nanjing Vocational University of Industry Technology, Nanjing, China
[2] School of Software, Nanjing University of Information Science & Technology, Nanjing, China

Currently, the challenges in utilizing PEC to process sensor data routing can be mainly divided into two aspects. On the one hand, due to the following factors, such as the diversities of user needs, the temporal differences in service access, and the limited coverage of an individual edge node, if a certain sensor data is cached on an edge node, it often leads to an increase in service latency, and even network congestion and service interruption. On the other hand, due to differences in computing power and dynamic changes in load conditions among edge nodes, some edge nodes may have excessive load, which greatly damages the quality of service. In the paper, we focus on the topic of ARPEC, specifically studying how to reduce data traffic, how to shorten user access latency, and how to improve the utilization of edge nodes in PEC. The contributions of this paper can be summarized as follows.

- Modeling adaptive routing optimization: We analyze the key goals of solution to routing optimization, and propose a sensor data routing model to reasonably route sensor data to the target edge nodes which provide caching services for the sensor data.
- Selection scheme for target edge node: We introduce a target edge node selection scheme to enhance user experience, which can adapt to the temporal dynamic characteristics of routing demands, and design a target edge node selection algorithm based on a grey linear regression combination prediction model.
- Construction and solution of MCMF graph: To improve the overall performance of PEC, we map sensor data routing to MCMF graph from a resource perspective, at the same time, propose an incremental MCMF algorithm based on BP, which reduces computational costs and supports dynamic deployment.

The remainder of this paper is organized as follows. In Related work section, we review the related work on routing strategies. An adaptive routing model in PEC is established in Adaptive routing model Section. In Target edge node section, we propose a target edge node selection scheme based on the relevant characteristics of PEC and IoT, and design a request message processing algorithm. In MCMF graph section, by analyzing the impact of edge node load on service quality, we propose a sensor data routing scheme based on MCMF graph that supports global optimal solution for multiple routing tasks, and design an incremental MCMF algorithm based on BP. We compare the experimental performance and analyze the results in Simulation and performance analysis section. Conclusion section concludes the paper.

Related work

The sensor data routing strategy of edge computing network in IoT is a hot research topic in academic research in recent years. According to the different metrics, it mainly involves two aspects: scheme for selecting target edge node and solution using MCMF model.

Scheme for selecting target edge node

In IoT, the edge nodes store a certain amount of sensor data. The quality of target edge node selection scheme will have a significant impact on system performance. The core of scheme is how to cache sensor data, that is, how to design caching scheme for sensor data.

LCE [11] is an intuitive caching decision scheme where data is cached by all nodes along the way back to the requester; however, it can easily lead to huge data redundancy. LCD [12] and MCD [13] are two approaches that improve the caching diversity. A new copy of the requested data will be cached only at the immediate node of the path in downward direction. The difference between them is that under MCD, the hit node will delete the requested data. However, both were proposed earlier for web caching to reduce cache redundancy. Prob [14] is a probabilistic caching version of LCE. However, the version is not flexible enough to determine whether to cache the arrived data based on a static probability. In short, the above strategies are mainly applied to Information Centric Networks and not specifically targeted at IoT.

Currently, there are some data caching solutions based on IoT. Y. Sellami et al. [15] proposed an architecture for a distributed fog caching solution for Content-Centric Networks. O. Serhane et al. [16] focused on an energy-aware caching placement scheme to maximize the energy-saving. G. Dhawan et al. [17] considered the trade-off between node energy, data freshness and cache occupancy, and proposed a candidate node selection for cache placement. N. Baltagiannis et al. [18] incorporated sensor faults and energy consumption in caching decisions. M. Amadeo et al. [19] designed a probabilistic Internet-scale caching design for IoT data. S. Tanted et al. [20] discuss the design, implementation and performance of a distributed caching & aggregation mechanism to handle the visualization of sensor data. However, the approaches are more focused on caching techniques for IoT-based compared to giving much attention to the placement and selection of caching nodes.

In summary, it can be seen that the caching schemes of Information Centric Networks have the problems of high pressure on data center transmission and computing, as well as high construction costs; the IoT-based caching schemes are good solutions for the systems that generate a large amount of sensor data flow. However, how to improve resource utilization and computational efficiency, as well as how to reduce user query latency, is still a topic that needs further research. This study extends the existing work by utilizing an adaptive routing optimization model with mechanisms for selecting target edge node to address the limitations of traditional routing models when applied to PEC. It also optimizes the selection scheme for target edge node, resulting in reduced load on edge node and user access latency.

Solution using MCMF model

In many scenarios, a large number of users access edge nodes at different times and locations [21]. Therefore, routing strategy typically cannot be designed for each user, but relies on dynamic changes in user query patterns and spatiotemporal dimensions. In addition, routing service often incurs significant transmission costs, which may lead to service interruption and overall service delay. At present, there are some studies using MCMF model to solve sensor data routing problems.

To minimize average transmission delay or flow cost in data centers, Akbar Majidi et al. [22] formulated this optimization problem as a constrained Markov decision process. By creating multiple paths between the source and destination peers, MaxFlowTCP [23] utilized SDN with traditional TCP to deliver maximum flow throughput in data centers. Through a logically centralized control plane, the processing-while-transmitting pattern [24] jointly determined the transmission path and placed subtasks adaptive to dynamic topology, and data distributions. To minimize the sum of heterogeneous costs of relay nodes, Y. Sun et al. [25] formulated the NWPTSTP for minimum cost relay node placement. However, the approaches are more focused on providing traffic management in Software-Defined Networks environments or in Wireless Sensor Networks, and don't involve the re-optimization of flow in PEC.

In addition, some studies have concentrated on BP-based MCMF methods [26], which mainly concentrate on how to utilize the parallel transmission of information between nodes in solving combinatorial optimization problems. G. Dhawan et al. [27] applied BP to discover globally optimal routes by performing low-complexity computations and exchanging messages with their neighbors. N. Baltagiannis et al. [28] showed how BP can be used to compute the throughputs of different links in the network given their access intensities. Ghafoor et al. [29] applied BP algorithm to compute a final belief about existence of primary user. Kakkasageri et al. [30] proposed multi-agent routing scheme for vehicular Ad hoc networks. However, one common challenge is the high time-consuming of message updates, which often leads to a low sensor data routing efficiency. Thus, there is a need for further research to develop optimized BP algorithm that can effectively minimize system overhead in PEC.

Drawing inspiration from these above researches, we focus on the overall performance of PEC utilizing an adaptive routing optimization model, and demonstrate how leveraging an incremental MCMF algorithm based on BP can reduce computational costs, thereby enhancing efficiency in routing sensor data.

Adaptive routing model

System model

A typical PEC scenario is shown in Fig. 1.

PEC consists of sensors, edge nodes, and users. Without affecting routing analysis, the system model omits gateways between sensors and edge nodes, as well as between users and edge nodes. There are two types of messages in PEC: user request and sensor data. Thus, it can be inferred that the overall network traffic arises from the superposition of these two types of messages. Assume that sensor data carries the corresponding information such as sensor identification and timestamp. A sensor transmits data to a directly connected edge node, which stores sensor data in the same size blocks. We refer to the data blocks in edge node as content. If an edge node caches a certain sensor data, then the edge node is referred to as the target edge node related to the sensor. Edge nodes can transmit user requests and sensor data between each other. A user obtains the contents of a target edge node by sending request messages. In Fig. 1, $S = \{s_1, s_2, s_3\}$, $EV = \{ev_1, ev_2, \ldots, ev_6\}$ and $U = \{u_1, u_2\}$ denote the sets of sensors, edge nodes and users, respectively. The set of target edge nodes related to s_1 for the users u_1 and u_2 is denoted by $TE(s_1, U) = \{ev_2, ev_5\}$. A label $\overset{Data}{\rightarrow}$ on an edge indicates the routing of sensor data. Meanwhile, a label $\underset{Request}{\rightarrow}$ expresses the routing of user request messages.

With the expansion of the scale of PEC scenarios, this system model exhibits highly dynamic characteristics. First, with the dramatic increase of sensors, the user request services are highly dynamic in temporal. Thus the temporal dynamic of user request services should be considered for routing optimization. Second, it is demonstrated that with the deepening of PEC computing, an unchanged topology in a PEC scenario can no longer meet complex application requirements. So, supporting dynamic deployment of nodes is another factor that cannot be ignored in routing optimization.

Optimization goals

Considering the dynamic requirements of user request services and network topology, the adaptive routing in

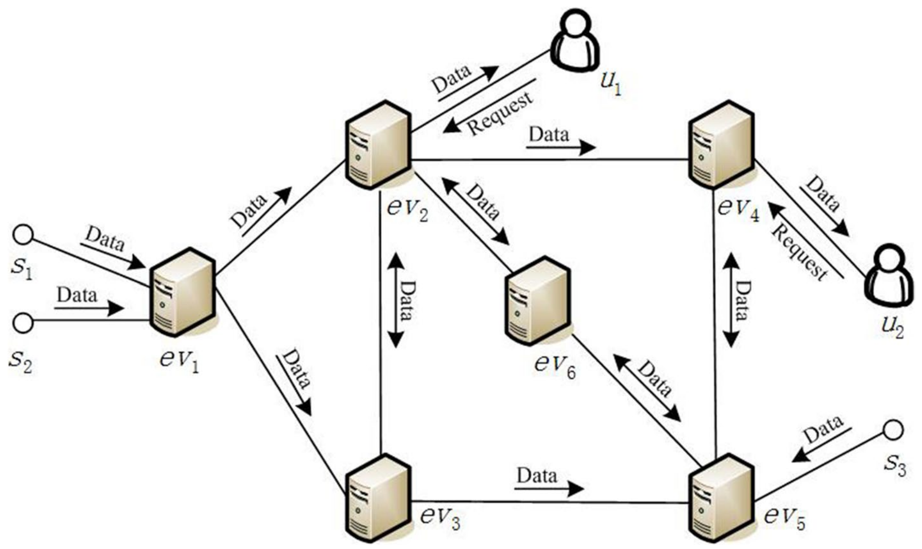

Fig. 1 A typical PEC scenario

the paper is defined as how to reasonably route sensor data to the edge nodes for caching in heterogeneous edge node scenarios at each time, that is, how to select the target edge node to achieve the optimal goal.

From the perspective of edge computing service providers and users, the main purpose of selecting target edge node is to reduce the traffic of entire network and the access delay of users, while improving the resource utilization of the edge node. To achieve this goal, the following measures can be taken. Firstly, In order to avoid the surge in the traffic of entire network caused by the propagation of sensor data in PEC, we reduce the number of visits to edge nodes directly connected to sensors. Secondly, we make the content as close to users as possible to reduce users' access delay. Thirdly, considering the performance differences of edge nodes, we deploy user request services to edge nodes that minimize the overall computational network scenario overhead as much as possible. Therefore, for selecting target edge node, three optimization goals are defined as follows.

(1) Goal 1: Minimize the load on edge nodes directly connected to sensors

The goal 1 can be calculated as

$$load(u,s) = \sum_{t \in T} sat(req(u,s,t), dl(\mathrm{s})), \tag{1}$$

where T is the running time of the network, *req*(u,s,t) is the request messages sent by user u to s at time t, *dl*(s) represents the edge node directly adjacent to s, and *sat*(r,*ev*) is a Boolean function to determine whether *ev* meets the request r. We define *load*(u,s) as the number of requests that edge node *dl*(s) satisfies u during time period T. The smaller the value of *load*(u,s) is, the lower the network traffic is.

(2) Goal 2: Minimize user access latency

The goal 2 can be calculated by the following expression.

$$delay(u,s,ev) = d(u,s,ev) \cdot \sum\nolimits_{t \in T} sat(req(u,ev,t), dl(s)). \tag{2}$$

We define *delay*(u,s,*ev*) as the ratio of the delay of sensor data transmission between user u and the edge node *ev* that satisfies the user's request after sending a request message to sensor s during time period T, and the delay between user u and the edge node *dl*(s). The smaller the value of *delay*(u,s,*ev*), the closer the edge node that satisfies the user u's request is to u. Therefore, *delay(u,s,ev)* can be used to indirectly reflect goal 2.

(3) Goal 3: Minimize system overhead

One of the factors affecting system overhead is the resource utilization of edge nodes. To describe the resource utilization of edge node, the weight value of edge node *ev*, denoted as w(*ev*), is designed in PEC. w(*ev*) is inversely proportional to the resource utilization of the edge node. Here, w(*ev*) is influenced by the factors such as the CPU and disk utilization of edge node, as well as the priority of content requested by user, that is, the higher the utilization of CPU and disk, as well as the priority of the requested content, the lower this value of w(*ev*) is.

The goal 3 can be formulated as follows.

$$cost(u,s) = w(dl(s)) \cdot load(u,s) + mW \cdot mD, \tag{3}$$

where $mW = (w(ev_1)w(ev_2)\cdots w(ev_n))$ is a 1*n matrix, given n is the number of edge nodes in the network. $mD = (delay(u,s,ev_1)\cdots delay(u,s,ev_n))\prime$ is a n*1 matrix. Taking into account the processing capacity of the edge nodes, *cost*(*u*,*s*) is defined as a numerical value that represents the ratio of the product of the edge node load of the content provider in goal.1 and the delay between *u* in goal.2 and the edge node *ev* that satisfies *u*'s request, to the delay between *u* and the content source provider.

As mentioned above, we construct the following adaptive routing optimization model.

$$min \sum_{\substack{u \in U \\ s \in S}} cost(u,s). \tag{4}$$

Considering the different processing capabilities of each edge node in the scenario of heterogeneous edge nodes, the target of the model is to solve for the values of the *req* function and *sat* function at a certain time, so as to minimize the value of formula 4.

Problem analyses

(1) Selection of target edge node

By observing the goal 1 and the goal 2, we find that *req*(*u*,*s*,*t*) and *req*(*u*,*ev*,*t*) are key parameters in the model, which can reduce the overall network traffic and user access latency. Thus, based on the value of the function *sat*, it can be determined whether the edge node has a message that meets the request. By summing the *sat* values, the activity of sensor messages at an edge node can be determined. Based on message activity, it is possible to better determine whether each edge node is a target edge node during the period. Therefore, the value of $\sum_{t\in T} sat(r,ev)$ is a key factor. In addition, in order to minimize user access latency, it is necessary to keep $\sum_{t\in T} sat(r,ev)$ as small as possible, that is, the number of hops between the target edge node and the user needs to be as small as possible. We proposes a target edge node selection scheme based on message activity and network topology, and implements a target edge node selection algorithm based on a grey linear regression combination prediction model.

(2) MCMF graph

By observing the goal 3, we find that *w*(*ev*) is another key parameter in the model. Furthermore, *w*(*ev*) can be quantified as the capacity of the path in the network graph. At present, the MCMF problem has been applied to various network routing scenarios and is usually characterized by graph models. By limiting the capacity and cost of paths in the graph, the global optimal solution for multiple routing tasks is sought. We apply it to message routing modeling in multi user, multi sensor, and multi query scenarios. Specifically, the two problems of users' demand for resources and the supply of resources in edge computing networks are transformed into the construction and solution of MCMF graph. In particular, considering the throughput and response delay of user requests in edge computing networks, an incremental MCMF optimization algorithm is implemented.

A MCMF graph for sensor data routing is shown in Fig. 2. The nodes marked with dashed circles, such as ev_2 and ev_5, are the target edge nodes of s_1. The node tv_{s_1} is the terminal node associated with s_1, which serves as the aggregation node for data from sensor s_1. tv_{s_1} is connected to u_1 and u_2 by edges, indicating that u_1 and u_2 have sent request messages to s_1. Sequence $s_1 \rightarrow ev_1 \rightarrow ev_2 \rightarrow u_1$ and sequence $s_1 \rightarrow ev_1 \rightarrow ev_3 \rightarrow ev_5 \rightarrow ev_4 \rightarrow u_2$ represent the two data forwarding paths related to s_1, respectively. Each edge is marked with a pair of cost/capacity values, where the cost indicates the cost of forwarding sensor messages through the edge, and the capacity means the ability of the edge to forward sensor messages. Considering that an edge node directly connected to a sensor should receive the sensor data as much as possible, the capacity of the edge is assigned infinity. Similarly, due to the fact that the terminal node directly connected to a user is uniformly added for the convenience of unified processing, the capacity of the edge is also assigned infinity.

Target edge node

Priority of edge node

In the following, based on the scheme of selecting target edge nodes, we first discuss three time-related metrics: message activity, user request probability, and query distance. And then, the priority is set for each edge node based on these measurements, and the edge node with the highest priority is selected as the target edge node to meet the dynamic changes in user service requests.

(1) Message activity

Since the types and quantities of user requests received by edge node are different, we can determine the activity of message by calculating the number of user requests. Therefore, message activity is defined as the ratio of the number of requests for that message to the total number of requests for all messages. Let r_s denote the messages generated by the sensor *s*. For the edge node *ev*, *act*(r_s,*t*,*ev*) denotes the activity of r_s at time *t*. The activity of r_s in the edge node *ev* can be denoted by *act*(r_s,*ev*). Thus, the following equation holds.

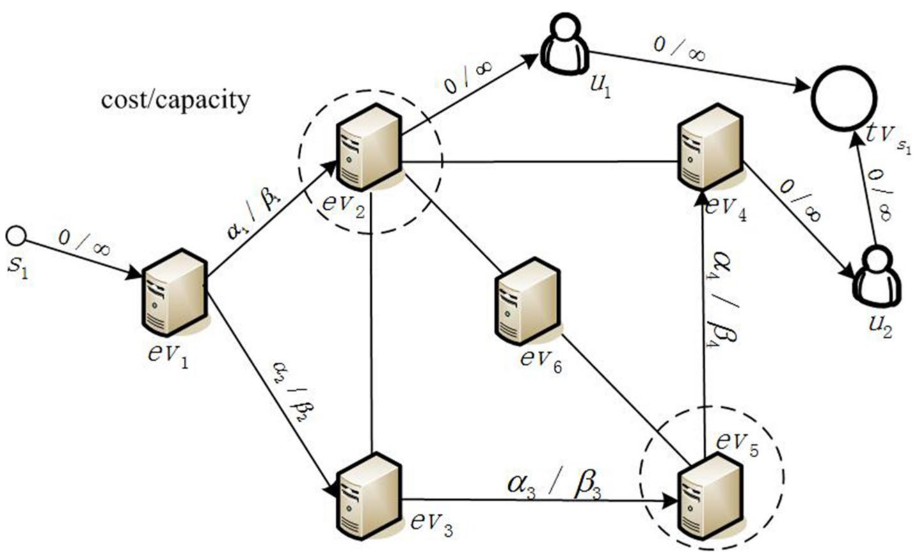

Fig. 2 A MCMF graph for sensor data routing

$$act(r_s, ev) = \sum_{t\in T} act(r_s, t, ev) = \sum_{t\in T} sat(r_s, ev). \tag{5}$$

(2) User request probability

Considering that a sensor generates new messages every once in a while, users are more inclined to request the latest information. Therefore, the novelty of messages can have an impact on the priority of the edge nodes. We use timestamps to distinguish messages generated by the same sensor at different times. So the larger the timestamp value is, the newer the message is. The probability of any user sending a request message to the edge node *ev* regarding the message *r* generated by sensor *s* at time t is denoted by $prob(r_s,t,ev)$.

(3) Query distance

In order to reduce latency, sensor messages should be cached as close to the edge node as possible to the user who sends the relevant request message. Therefore, we take the network topology into account when calculating the priorities of edge node. The query distance is denoted by $ah(r_s,t,ev)$, which is defined as the average number of edge nodes on the relevant paths. Here, the relevant path refers to the path that the message r_s generated at time *t* routes from the current edge node *ev* to the user node that send the corresponding request message.

(4) Selecting target edge node

Message activity and user request probability are related to the usefulness of the cached sensor data, and query distance is a metric of user access latency. In summary, for the sensor message r_s generated at time *t*, the priority of the edge node *ev* can be obtained by

$$p(r_s, t, ev) = act(r_s, t, ev) \cdot prob(r_s, t, ev)/ah(r_s, t, ev). \tag{6}$$

Similarly, for the sensor message r_s, the priority of the edge node *ev* can be calculated as

$$p(r_s, ev) = \alpha_{ev} \sum_{t\in T} p(r_s, t, ev)/ \sum_{t\in T, s\in S} p(r_s, t, ev). \tag{7}$$

Among them, the coefficient α_{ev} denotes an adjustment factor, and its value can be appropriately predicted according to the application scenario. In the paper, we let $a=1$. Generally speaking, the higher the activity of message is, the higher probability of user's request is; and the smaller the query distance is, the higher priority of edge nodes.

Prediction of priority

In PEC, sensors constantly generate new data, and users frequently have new requests. Especially when edge nodes receive new requests, the three metric values calculated by using the method in the previous section will have a significant deviation from the subsequent stable values. This results in a significant difference in the priority of edge nodes between the early and the subsequent stable stages. In order to overcome the above impact, this section predicts the priority of edge nodes that receive new requests, so that the nodes can obtain more accurate priorities earlier. Due to the characteristics of grey system, such as no need for a large number of samples, and a good short-term prediction performance, a grey system prediction model for predicting priority is proposed.

Assume that a user has a request for a new message $r_{s'}$ from sensor s, where $r_{s'}$ is an earlier message than r_s. For predicting the priority of ev regarding $r_{s'}$, the priority sequence of ev regarding r_s can be quantified as $mP^{(0)} = (p(r_s, t_1, ev), p(r_s, t_2, ev), \ldots, p(r_s, t_n, ev))$, and then, the grey system prediction model, which is donoted by $p^{(1)}(r_s,t_k,ev)$, is used to estimate the predicted value $p(r_s, t_{n+1}, ev)$. The details of modeling can be found as follows.

Step 1. Accumulate the priority sequence $mP^{(0)}$ to obtain a new sequence, i.e. $mP^{(1)} = (p^{(1)}(r_s, t_1, ev), p^{(1)}(r_s, t_2, ev), \ldots, (r_s, t_n, ev))$. The accumulation formula can be expressed as

$$p^{(1)}(r_s, t_k, ev) = \sum\nolimits_{i=1}^{k} p(r_s, t_i, ev), k = 1, 2, \ldots, n. \tag{8}$$

Step 2. Generate the adjacent difference sequence $mZ = (z(1), z(2), \ldots, z(n-1))$, where $z(k)$ follows the below expression.

$$z(k) = p^{(1)}\left(r_s, t_{k+1}, ev\right) - p^{(1)}(r_s, t_k, ev), k = 1, 3, \ldots, n-1. \tag{9}$$

Step 3. Calculate the estimated values of $\widehat{v}$, $\widehat{c}_1$, $\widehat{c}_2$, and $\widehat{c}_3$ according to the following formulae.

$$\begin{aligned} \widehat{v} = aver(&\sum\nolimits_{m=1}^{n-3} \sum\nolimits_{k=1}^{n-2-m} (ln(z(k+m+1) - z(k+1)) \\ &- \ln(z(k+m) - z(k)))), \end{aligned} \tag{10}$$

$$mC = \left(mA' \cdot mA\right)^{-1} \cdot mA' \cdot (mP^{(1)})' \tag{11}$$

where aver is a function for calculating the average value, $mC = \begin{pmatrix} c_1 & c_2 & c_3 \end{pmatrix}'$, and $mA = \begin{pmatrix} e^{\hat{v}} & 1 & 1 \\ e^{2\hat{v}} & 2 & 1 \\ \vdots & \vdots & \vdots \\ e^{n\cdot\hat{v}} & n & 1 \end{pmatrix}$

Step 4. Calculate the estimated value of $p,^{(1)}(r_s t_k, ev)$ by using the following prediction model.

$$p^{(1)}(r_s, t_k, ev) = \widehat{c}_1 \cdot e^{\widehat{v}k} + \widehat{c}_2 \cdot k + \widehat{c}_3, k = 2, \cdots, n+1. \tag{12}$$

Step 5. Restore the results of step 4.

$$\widehat{p}^{(0)}\left(r_s, t_{k+1}, ev\right) = \widehat{p}^{(1)}\left(r_s, t_{k+1}, ev\right) - \widehat{p}^{(1)}(r_s, t_k, ev), k = 1, 2, \cdots n, \tag{13}$$

where $\widehat{p}^{(0)}\left(r_s, t_{k+1}, ev\right)$ is the predicted priority of ev regarding $r_{s'}$.

Algorithm for processing request messages

The priority adjustment of edge node is accompanied by the processing of user request messages by the edge node. According to the discussions of priority predicting, the process of request message processing algorithm based on a grey system prediction model includes the following essential steps. When receiving a user request message, the edge node updates the values of these metrics: message activity, user request probability, and query distance. If the request is a new request, it is necessary to predict the priority of the edge node regarding this request. Finally, the edge node refreshes the user request message and recalculates the priority corresponding to each request. The details can be found in Algorithm 1.

Lines 3-6 are the setting of measurements and the return of results to users when the edge node detect user request messages; lines 7-8 are the forwarding of request messages; lines 9-12 predict the priority of the edge node when dealing with new request messages; The 13th line recalculates the priority of the edge node after changes in these measurements.

Algorithm 1. Request message processing

```
Input: request message req, edge node ev
Output: sensor data sd
1. Procedure RequestHandle(req,ev)
2.   While detecting a request Do
3.     (actv, probv, ahv)=Indicatorcalculating(req,ev);
4.     If Targetculating(req, ev) Then
5.       reply(req.user, sd);
6.     End Then
7.     Else
8.       Forward(req);
9.     End Else
10.    If Isnewrequest(req, ev) Then
11.      GreyPre(0, ev);
12.    End Then
13.    priorityculating(ev);
14. END While
15.END
```

In Algorithm 1, the edge node *ev* maintains operations on its content based on a content block statistics table, which consists of user request messages, sensor data messages, and three metrics related to optimization goals. The statistics table is an important data structure that supports the selection of target nodes. Assume the length of the statistical table is *m*, and the number of sensors related to the content is *l*. Let $C_{req,s}$ denote a set of all content blocks with the same sensor identifier *s* as that of *req*, where the timestamps in the set are earlier than *req*. Let $|C_{req,s}|=n$. For the maintenance operation of the statistical table, the time complexity analysis is as follows. There are only two procedures, i.e., as *GreyPre* and *priorityculating*, involved in operating the statistical table. *GreyPre* includes the steps such as traversing $C_{req,s}$, sorting $C_{req,s}$ in chronological order, and calculating according to formulas (8, 9, 10, 11, 12 and 13). The time complexity of *GreyPre* is $O(n\log_2 n)$. Similarly, *priorityculating* the time complexity of *GreyPre* is $O(m + l \cdot n \cdot log_2 n)$. In conclusion, the time complexity of Algorithm 1 is $O(m\log_2 n)$.

MCMF graph

Construction of MCMF graph

We mainly consider the performance parameters such as reducing network traffic, shortening user access latency, and improving the utilization of edge nodes. This section discusses the question that how to characterize the above parameters from a resource perspective and how to convert them into the assignment problem of graph capacity and cost.

For a given graph $G = (V, DE, CA, CO)$, among them, *V* is a node set that represents entities in PEC, including the sensor *S*, the edge node *EV*, the user *U*, and the terminal node *TV*; *DE* is a directed edge set, which indicates whether a data forwarding task can be mapped to a corresponding entity; *CA* denotes a set of capacities on edge, identifying the entity's supply capacity; *CO* is a cost set that represents the mapping effect between sensor data forwarding tasks and entities.

Definition 1

For $G, \exists s \in S \bigwedge \exists v_i, v_j \in V \bigwedge (v_i, v_j) \in DE$, to analyze the flow in PEC, we let $out(s, v_i, v_j)$ denote the flow of the edge $<v_i,v_j>$ with respect to *s*, which can be calculated as follows.

- If $v_i \in EV, (s, v_i) \in DE$, then $out(r_s, s, v_i) = 0$,
- If $v_i, v_j \in EV$, then $out(r_s, v_i, v_j) = 1$,
- If $v_i \in EV, v_j \in U$, then $out(r_s, v_i, v_j) = 0$,
- If $v_i \in U, v_j \in TV$, then $out(r_s, v_i, v_j) = 0$.

Similarly, the flows of $<v_i,v_j>$ and G are expressed as $out(v_i, v_j) = \sum_{s \in S} out(r_s, v_i, v_j)$ and $out(G) = \sum_{(v_i,v_j) \in DE} out(r_s, v_i, v_j)$, respectively.

Definition 2

For $G, \exists s \in S \bigwedge \exists v_i, v_j \in V \bigwedge (v_i, v_j) \in DE$, the cost of the edge $<v_i,v_j>$ with respect to *s*, i.e. $co(s, v_i, v_j)$, can be calculated as follows.

- If $v_i \in EV, <s, v_i> \in DE$, then $co(r_s, s, v_i) = 0$,
- If $v_i, v_j \in EV$, then $co(r_s, v_i, v_j) = w(v_j) \cdot \frac{1}{p(r_s, ev)}$. The cost of forwarding data from v_i to v_j is related to $p(r_s, t, v_j)$ and $w(v_j)$, in the paper, it is specifically expressed as $w(v_j) \cdot \frac{1}{p(r_s, ev)}$,
- If $v_i \in EV, v_j \in U$, then $co(r_s, v_i, v_j) = 0$,
- If $v_i \in U, v_j \in TV$, then $co(r_s, v_i, v_j) = 0$.

In the same way, $co(v_i, v_j) = \sum_{s \in S} co(r_s, v_i, v_j)$ and $co(G) = \sum_{(v_i,v_j) \in DE} co(v_i, v_j)$ are the costs of $<v_i,v_j>$ and G.

Definition 3

For $G, \exists v_i, v_j \in V \bigwedge (v_i, v_j) \in DE$, then $ca(v_i, v_j)$ represents the capacity of the edge $<v_i,v_j>$, which can be obtained by the following expressions.

- If $\exists s \in S \bigwedge \exists v_i \in EV \bigwedge (s, ev) \in DE$, then $ca(s, v_i) = \infty$,
- If $ev_i, ev_j \in EV$, then $ca(ev_i, ev_j) = min(w(ev_i) \cdot \sum_{\substack{u \in U \\ s \in S}} delay(u, s, ev_i), w(ev_i) \cdot \sum_{\substack{u \in U \\ s \in S}} delay(u, s, ev_i))$,
- If $v_i \in U \bigwedge v_j \in TV$, then $ca(r_s, v_i, v_j) = 0$.

Solution of MCMF graph

(1) Formal description of problem solving

In this paper, one of our purposes is improving the overall performance of PEC. So we formulate the MCMF problem for graph *G* as follows.

$$\min \sum_{(v_i,v_j) \in DE} out(v_i, v_j) \cdot co(v_i, v_j), \tag{14}$$

$$s.t. C1 : \sum_{(v_i,v_j) \in DE} f((v_i, v_j)) - \sum_{(v_j,v_i) \in DE} f(v_j, v_i) = b(v_j), \tag{15}$$

$$C2 : 0 \le out(v_i, v_j) \cdot co(v_i, v_j) \le ca(v_i, v_j), \forall (v_i, v_j) \in DE. \tag{16}$$

As mentioned above, the node that satisfies the condition $b(v_j) > 0$, and $b(v_j) < 0$ is called a source node, and an aggregation node, respectively; the flow that satisfies

$\sum_{v \in EV} b(v) = 0$ is called a feasible flow. Under the premise of satisfying formulae (14, 15 and 16), we maintain the minimum cost, and iteratively search for the maximum flow until the optimization conditions are met.

In a PEC environment, there will be certain changes in PEC, such as the number of user requests, the access costs, the load capacity of edge nodes and the number of edge nodes. However, the structural changes in the graph associated with PEC are relatively small within every two adjacent time periods. Therefore, we propose an incremental MCMF algorithm based on BP. By caching and reusing the last solution, only local operations are required to obtain the global optimal solution of the graph.

(2) Algorithm for solving MCMF graph

BP algorithm is message-transmission algorithm based on factor graph model. By updating the information of nodes, BP algorithm updates the state of an entire network until the network stabilizes. The efficiency of message transmission plays a crucial role in BP algorithm. This section discusses the incremental MCMF algorithm based on BP, which reduces the network and computational costs and supports dynamic deployment by only sending the changed messages.

In a factor graph, there are two types of nodes: variable node x_i and constraint node $f(a)$. Correspondingly, there are two types of messages on undirected edge (i,α) at time t, namely message $m(i,a,x_i,t)$ sent by x_i to $f(a)$, and message $g(a,i,x_i,t)$ sent by $f(a)$ to x_i. The message description based on increment is as follows.

$$\Delta m(i,a,x_i,t) = \sum_{k \in N(i) \backslash a} \Delta m(k,i,x_i,t-1), \tag{17}$$

$$m(i,a,x_i,t) = m(i,a,x_i,t-1) + \Delta m(i,a,x_i,t), \tag{18}$$

$$\Delta g(a,i,x_j,t) = \sum_{j \in N(a) \backslash i} \Delta m\left(j,a,x_j,t\right), \tag{19}$$

$$g(a,i,x_j,t) = g(a,i,x_j,t-1) + \Delta g(a,i,x_j,t), \tag{20}$$

where $k \in N(i) \backslash a$ denotes the set of constraint nodes connected to x_i except for $f(a)$, $j \in N(a) \backslash i$ denotes the set of variable nodes connected to $f(a)$ except for x_i, and let $\Delta m(i,a,x_i,0) = 0, \Delta m(k,i,x_i,0) = m(k,i,x_i,0)$,$g(a,i,x_j,0) = 0$.

For graph G, f denotes its minimum cost flow, and graph $G' = (V', DE', CA', CO')$ denotes the graph after its local changes. By further optimizing the minimum cost flow f, the incremental minimum cost flow algorithm obtains the minimum cost flow f 'of graph G' according to the factor graph increments and the message increments. The details can be found in Algorithm 2.

Lines 2-3 generate the factor graphs and the minimum cost flow for the initial graph G; lines 4-8 detect the state change events, update the graph structure when events occur, and solve the updated graph; lines 11-27 are the main body of the incremental minimum cost flow algorithm based on belief propagation, with the main idea of calculating the minimum cost eigenvalues while ensuring the maximum flow constraint; lines 30-35 are incremental iteration operations, which specifically include the calculation of the factor graph increments and the message increments.

Algorithm 2. Incremental MCMF based on BP

```
Input: graph $G$
Output: the updated MCMF of graph $G$
1. Procedure IncrementalBPMCF($G$)
2.   ($G_f$,nill)=FGConverting($G$,nill)
3.   Init $f$=BasicBPMCF($G_f$)
4.   While detecting an event Do
5.     ($G'$,$\Delta G$)=Updategraph($G$, Events)
6.     $f'$=BasicBPMCF($G'$, $\Delta G$, $f$)
7.     $f$=$f'$
8.   END While
9. END
10.Procedure BasicBPMCF($G$,$\Delta G$,$f$)
11.   If $f$=nill Then
12.     $t$=0
13.     $m(i,a,x_i,0) = 0, \Delta m(i,a,x_i,0) = 0$
14.     $\Delta m(k,i,x_i,0) = m(k,i,x_i,0), \forall k \in N(i) \backslash a$
15.     $g(a,i,x_j,0) = 0$
16.   End If
17.   Else
18.     $(G'_f, \Delta m(i,a,x_i,t), \Delta g(a,i,x_j,t))$= IncrementaMsglUpdating($G$,$\Delta G$)
19.     $m(i,a,x_i,t) = m(i,a,x_i,t-1) + \Delta m(i,a,x_i,t)$
20.     $g(a,i,x_j,t) = g(a,i,x_j,t-1) + \Delta g(a,i,x_j,t)$
21.   End Else
22.     $t$=$t$+1
23.     For $x_i \in V(G_f)$ Do
24.       $b_i(x_i) = beliefcalculating(m(i,a,x_i,t), g(a,i,x_j,t))$
25.     End For
26.     $f' = estimatecalculating\left(arg\,min(b_1(x_1)), \dots, arg\,min(b_{|V(G_f)|}\left(x_{|V(G_f)|}\right)\right))$
27.     Return $f'$
28.End
29.Procedure IncrementalMsgUpdating($G$,$\Delta G$)
30.   ($G_f$, $\Delta G_f$)= FGConverting($G$,$\Delta G$)
31.   For $x_i \in V(\Delta G_f)$ Do
32.     $\Delta m(i,a,x_i,t) = \sum_{k \in N(i) \backslash a} \Delta m(k,i,x_i,t-1)$
33.     $\Delta g(a,i,x_j,t) = \sum_{j \in N(a) \backslash i} \Delta m(j,a,x_j,t)$
34.   End For
35.   Return($\Delta G_f, \Delta m(i,a,x_i,t), \Delta g(a,i,x_j,t)$)
36.End
```

For BP algorithm, the schedule for updating messages is crucial to its convergence. In Algorithm 2, we address the question of how to schedule messages for asynchronous propagation to attempt to reach convergence. Procedure *IncrementalMsgUpdating* leverages a scheduling scheme that selects all incremental messages to update at a time. Additionally, an incremental update approach is introduced to accelerate the computation of message in Procedure *BasicBPMCF*.

Simulation and performance analysis

Experimental settings

PeerSim [31] is an event simulator for large-scale distributed systems that supports P2P network architecture. To evaluate the performance of the proposed adaptive routing strategy, we conducted relevant simulation experiments on PeerSim.

The experiment use a cube-connected cycle graph [32] with $4*2^4=64$ nodes as the network topology structure. Each node in the graph denotes an edge node. Given that each edge node is connected to 10 sensors and 10 users, there are a total of 640 different sensors and 640 different users in the network. Each edge node can provide data generated by the connected sensor nodes to users, and the farthest location for a request message to receive a response is at the edge node connected to the corresponding sensor node.

The experimental parameters are set as follows. The bandwidth and latency between adjacent edge nodes are 1 Mb/second and 1 ms, respectively. A sensor generates a new message with a size of 10 MB every 1 s. The probability of users requesting different sensor messages is the same. And the probability of users requesting messages generated by the same sensor at different times follows *Zipf* distribution, where let $C = 1, 0.8 \leq \alpha \leq 1.5$. Each user randomly requests 20 different sensor messages at the rate of 1 request/second. Each edge node caches 2048 sensor messages and can accommodate 500 request messages. In each experiment, we take 5 samples with an interval of 60 s between each sampling. A total of 10 experiments were conducted, and the results were averaged.

According to the above settings, the total number of user request messages during the 5 sampling periods is $tnr=192000$.

Performance metrics

By analyzing the formula (1, 2 and 3) involved in the optimization goals of the adaptive routing model, we can conclude that $load(u,s)$ and $delay(u,s,ev)$ are key factors that affect the performance of the system model. To evaluate the performance of different routing schemes, the following two performance evaluation metrics based on $load(u,s)$ and $delay(u,s,ev)$ are considered.

- Average access latency, $adelay = \frac{1}{tnr} \cdot \sum_{t \in T} delay(u,s,ev)$, which denotes the average delay spent from an user sending a request to receiving the corresponding sensor message.
- Average edge node load, $aload = \frac{1}{tnr} \cdot \sum_{t \in T} load(u,s)$, which denotes the average number of times that edge nodes directly connected to sensors provide services to users.

Based on these indicators, we can analyze both the efficiency of caching services provided by target edge nodes in ARPEC and the effectiveness of target edge nodes in alleviating service pressure on edge nodes directly connected to sensors.

Performance testing & analysis

(1) Performance of request message processing algorithm

We adjust *Zipf* probability parameter α of request message to examine the performance trend of the request message processing algorithm. Because LCE, LCD, and Prob0.5 are three typical target node selection strategies that range from simple to complex, we compare the performance of ARPEC with that of LCE, LCD, and Prob0.5. The performance changes are shown in Fig. 3.

Figure 3a and b show that there are certain differences of the strategies and ARPEC outperforms these compared strategies in the two metrics. This is because user's requests for high-priority edge nodes become more frequent as the parameter α increases. Due to the lack of sensitivity of LCE, LCD, and Prob0.5 to changes in message activity and user's requests, performance improvement is limited. As message activity increases, sensor messages in edge nodes become more concentrated. By increasing the priority of the corresponding edge nodes, ARPEC can cache sensor messages on the reasonable edge nodes.

(2) Performance of incremental MCMF algorithm

The experimental parameters are set as follows. First, by assigning a value greater than 1 to a, it indicates that users are more concerned about several sensors. On the contrary, the distribution of user requests is relatively uniform. So α reflects user's preference for sensor requests. Without loss of generality, let $\alpha=1$. Second, in each sampling period, the number of nodes with changes is less than 12; and the change pattern of nodes follows a uniform distribution. In Fig. 4, the trends of the incremental BP-based MCMF algorithm in terms of performance measurements are examined by adjusting the number of changing nodes, and the results are compared with the *BasicBPMCF* algorithm.

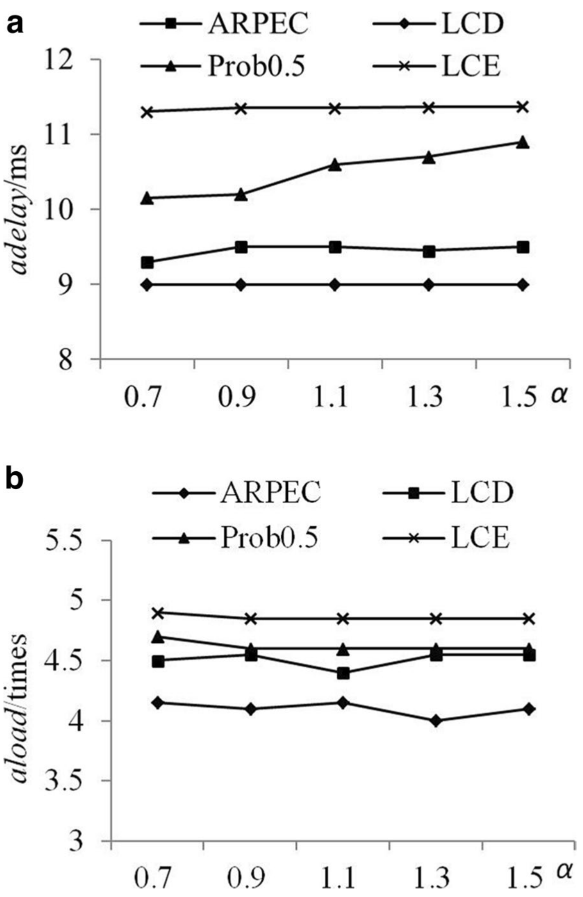

Fig. 3 Performance changes with *α* under different strategies. **a** Average access latency. **b** Average edge node load

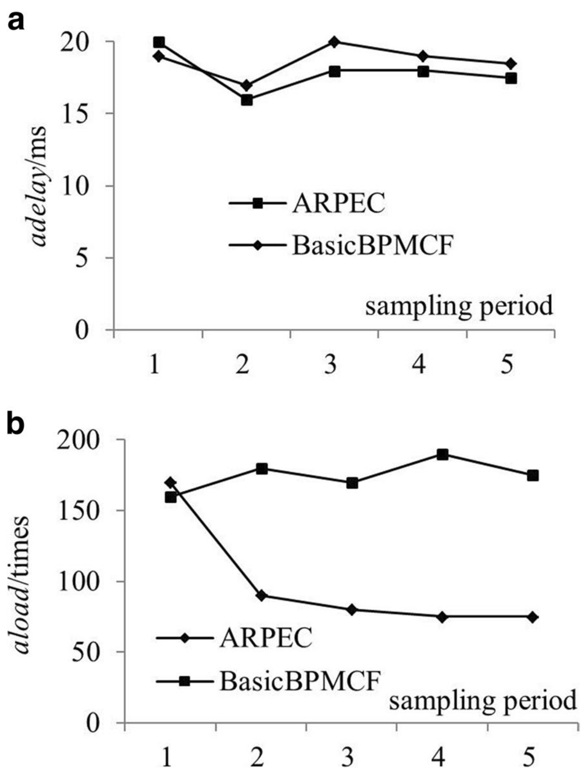

Fig. 4 Performance changes under different strategies. **a** Average access latency. **b** Average edge node load

The average access latency difference between the two strategies shown in Fig. 4a is not significant, which indicates that the solution results of the two strategies are basically consistent. However, as shown in Fig. 4b, the efficiency of the incremental MCMF algorithm has been greatly improved, with an edge node load of approximately 50% of that of BasicBPMCF. This indicates that the incremental MCMF algorithm reduces network costs. This is because the incremental MCMF algorithm saves time in solving the maximum flow. Meanwhile, because the data in the network does not change much, the MCMF in the previous stage may already be very close to the optimal flow in this stage, and only a small amount of incremental information needs to be processed to achieve the optimal feasible flow.

Analysis of ARPEC application scenarios

ARPEC uses a grey model to predict the priority of edge nodes. For cases where priority data samples change rapidly, the sample quality may not be particularly good and the response speed may not meet user requirements. We can improve sample quality by adjusting the length of the priority sequence. However, due to the unpredictable statistical characteristics of the incoming user requests and sensor data, *GreyPre* may adjust the sample length multiple times and results in higher latency. Therefore, ARPEC is not suitable for applications with significant instantaneous changes in IoT and high requirements for real-time computing.

The requirements for updating node information in ARPEC should be met. Users can set the frequency of updating node information. If the frequency is higher, the number of adjustments to the minimum cost flow will increase. Due to the limited resources of edge nodes, in the worst case, the adjustment of the minimum cost flow still cannot meet the user's requirements for network traffic. So, it is considered that ARPEC cannot provide support for MCMF at the given frequency of updating node information.

Conclusion

In this paper, we propose an adaptive routing strategy in PEC, which can adapt to the temporal dynamic characteristics of routing demands and supports dynamic deployment. In order to improve the user experience and give full play to the overall performance of Edge Cloud, a routing optimization model is described from the perspectives of network traffic, user access delay, and edge node resource utilization. We implement two algorithms, the processing request messages algorithm based on grey prediction model, and the incremental MCMF algorithm based on BP algorithm. The evaluation results show that ARPEC can effectively improve PEC transmission performance and user experience. In the future, our research will focus on the following two aspects: the optimization technology of target node parameters based on machine learning, and the merging and filtering mechanisms to reduce the complexity of solving graph.

Acknowledgements
We sincerely thank the editors and reviewers for their valuable comments on this paper.

Authors' contributions
B.D. and JH. C. wrote the main manuscript text and all authors reviewed the manuscript.

Funding
This work was supported by the industrial big data application technology project of excellent scientific and technological innovation teams in universities in Jiangsu Province, China.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 7 October 2023 Accepted: 21 December 2023
Published online: 10 January 2024

References

1. Xu XL, Gu J, Yan HZ, Liu WT, Qi LY, Zhou XK (2023) Reputation-aware supplier assessment for blockchain-enabled supply chain in industry 4.0. IEEE Transact Industr Inform 19(4):5485–5494
2. Wang G, Li C, Huang Y, Wang X, Luo Y (2022) Smart contract-based caching and data transaction optimization in mobile edge computing. Knowl Based Syst. 252:109344
3. Yan HZ, Bilal M, Xu XL, Vimal S (2022) Edge server deployment for health monitoring with reinforcement learning in internet of medical things. IEEE Transact Comput Soc Syst. https://doi.org/10.1109/TCSS.2022.3161996
4. Janani T, Brindha M (2022) SEcure Similar Image Matching (SESIM): an improved privacy preserving image retrieval protocol over encrypted cloud database. IEEE Trans Multimedia 24:3794–3806
5. Xu XL, Liu ZJ, Bilal M, Vimal S, Song HB (2023) Computation offloading and service caching for intelligent transportation systems with digital twin. IEEE Trans Intell Transp Syst 23(11):20757–20772
6. Mara GC, Rathod U, Shreyas RRG, et al (2020) CRUPA: collusion resistant user revocable public auditing of shared data in cloud. J Cloud Comput 9(1). https://doi.org/10.1186/s13677-020-00188-5
7. Li Z, Xu XL, Hang T, Xiang HL, Cui Y, Qi LY, Zhou XK (2022) A knowledge-driven anomaly detection framework for social production system. IEEE Transact Comput Soc Syst https://doi.org/10.1109/TCSS.2022.3217790
8. Zhang Y, Yu H, Zhou W, Man M (2023) Application and research of IoT architecture for End-Net-Cloud Edge computing. Electronics 12(1):1. https://doi.org/10.3390/electronics12010001
9. Xu XL, Tang SZ, Qi LY, Zhou XK, Dai F, Dou WC (2023) CNN partitioning and offloading for vehicular Edge networks in Web3. IEEE Commun Mag 61(8):36–42
10. Karagiannis V, Schulte S (2021) edgeRouting: using compute nodes in proximity to route IoT data. IEEE Access 9:105841–105858
11. Dutta N, Patel SK, Faragallah OS, Baz M, Rashed ANZ (2022) Caching scheme for information-centric networks with balanced content distribution. Int J Commun Syst 7:35
12. Fan C, Shannigrahi S, Papadopoulos C, Partridge C (2020) Discovering in-network Caching Policies in NDN Networks from a Measurement Perspective. ICN '20: 7th ACM Conference on Information-Centric Networking. pp 106–116
13. Laoutaris N, Che H, Stavrakakis I (2006) The LCD interconnection of LRU caches and its analysis. Performance Evaluation 63(7):609–634
14. Laoutaiis N, Syntila S, Stavrakakis L (2004) Meta algorithms for hierarchical web caches. IEEE International Conference on Performance, Computing, and Communications. pp 445–452
15. Sellami Y, Jaber G, Lounis A (2022) Distributed Fog-based Caching Solution for Content-Centric Networking in IoT. 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC). pp 493–494
16. Serhane O, Yahyaoui K, Nour B, Moungla H (2021) Energy-aware Cache Placement Scheme for IoT-based ICN Networks. ICC 2021-IEEE International Conference on Communications. pp 1–6
17. Dhawan G, Mazumdar AP, Meena YK (2022) CNCP: A Candidate Node Selection for Cache Placement in ICN-IoT. 2022 IEEE 6th Conference on Information and Communication Technology (CICT). pp 1–6
18. Baltagiannis N, Kapetanidou IA, Tsaoussidis V (2023) EFPCaching: Energy-aware and Fault-tolerant Probabilistic Caching of popular IoT content in ICN. NOMS 2023–2023 IEEE/IFIP Network Operations and Management Symposium. pp 1–4
19. Amadeo M, Campolo C, Ruggeri G, Molinaro A (2022) Beyond Edge Caching: freshness and popularity aware IoT data caching via NDN at internet-scale. IEEE Trans Green Commun Netw 6(1):352–364
20. Tanted S, Agarwal A, Mitra S, Bahuman C, Ramamritham K (2020) Database and caching support for adaptive visualization of large sensor data. Proceedings of the 7th ACM IKDD CoDS and 25th COMAD. pp 98–106
21. Li Z, Xu XL, Cao XF, Liu WT, Zhang YW, Chen DH, Dai HP (2022) Integrated CNN and federated learning for COVID-19 detection on chest x-ray images. IEEE/ACM Trans Comput Biol Bioinf. https://doi.org/10.1109/TCBB.2022.3184319
22. Majidi A, Gao X, Zhu S, Jahanbakhsh N, Chen G (2019) Adaptive routing reconfigurations to minimize flow cost in SDN-based data center networks. International Conference on Parallel Processing. pp 1–10
23. Norah SBS, Mohammed JFA (2020) Utilizing SDN to Deliver Maximum TCP Flow for Data Centers. Proceedings of the 3rd International Conference on Information Science and Systems (ICISS'20). pp 181–187
24. Li X, Tang F, Zhu Y, Fu L, Yu J, Chen L, Liu J (2022) Processing-while-transmitting: cost-minimized transmission in SDN-based STINs. IEEE/ACM Trans Netw 30(1):243–256
25. Sun Y, Rehfeldt D, Brazil M, Thomas D, Halgamuge S (2020) A physarum-inspired algorithm for minimum-cost relay node placement in wireless sensor networks. IEEE/ACM Trans Networking 28(2):681–694
26. Palmieri FAN, Pattipati KR, Gennaro GD, Fioretti G, Verolla F, Buonanno A (2022) A unifying view of estimation and control using belief propagation with application to path planning. IEEE Access 10:15193–15216
27. Badiu MA, Saad D, Coon JP (2021) Self-organization scheme for balanced routing in large-scale multi-hop networks. J. Phys A Math Theor 54:045001
28. Kai CH, Liew SC (2012) Applications of belief propagation in CSMA wireless networks. IEEE/ACM Trans Netw 20(4):1276–1289
29. Ghafoor H, Koo I (2017) Software-defined networking-based cognitive routing protocol for vehicular ad hoc networks. 2017 International Conference on Circuits, System and Simulation (ICCSS). pp 162–166
30. Kakkasageri MS, Sataraddi MJ, Chanal PM, Kori GS (2017) BDI agent based routing scheme in VANETs. 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET). pp 129–133

31. Xie RC, Wang ZY, Yu FR, Huang T, Liu YJ (2021) A novel identity resolution system design based on Dual-Chord algorithm for industrial Internet of Things. Sci China Inf Sci 64(8):182301
32. Han ZJ, Fu YY, Du XX (2022) Embedding Cube-Connected Cycles into Grid Network for Minimum Wirelength. EEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'21). pp 461–465

Publisher's Note