A Dynamic Service Migration Mechanism in Edge Cognitive Computing

MIN CHEN, School of Computer Science and Technology, Huazhong University of Science and Technology, China and Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, China
WEI LI, School of Computer Science and Technology, Huazhong University of Science and Technology, China
GIANCARLO FORTINO, University of Calabria, Italy
YIXUE HAO and LONG HU, School of Computer Science and Technology, Huazhong University of Science and Technology, China
IZTOK HUMAR, University of Ljubljana, Faculty of Electrical Engineering, Slovenia

Driven by the vision of edge computing and the success of rich cognitive services based on artificial intelligence, a new computing paradigm, edge cognitive computing (ECC), is a promising approach that applies cognitive computing at the edge of the network. ECC has the potential to provide the cognition of users and network environmental information, and further to provide elastic cognitive computing services to achieve a higher energy efficiency and a higher Quality of Experience (QoE) compared to edge computing. This article first introduces our architecture of the ECC and then describes its design issues in detail. Moreover, we propose an ECC-based dynamic service migration mechanism to provide insight into how cognitive computing is combined with edge computing. In order to evaluate the proposed mechanism, a practical platform for dynamic service migration is built up, where the services are migrated based on the behavioral cognition of a mobile user. The experimental results show that the proposed ECC architecture has ultra-low latency and a high user experience, while providing better service to the user, saving computing resources, and achieving a high energy efficiency.

CCS Concepts: • **Networks** → **Network services**; • **Computer systems organization** → *Distributed architectures*; • **Theory of computation** → Design and analysis of algorithms;

This work is supported by the National Key R&D Program of China (2017YFE0123600, 2018YFC1314605), the China National Natural Science Foundation (No. 61802139, No. 61802138), the National Natural Science Foundation of China (under Grant No. U1705261), and Director Fund of WNLO. Dr. Humar would like to acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0246).

Authors' addresses: M. Chen, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074, China and Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, 430074, China; email: minchen@ieee.org; W. Li, Huazhong University of Science and Technology, School of Computer Science and Technology, Wuhan, 430074, China; email: weiliepic@hust.edu.cn; G. Fortino, University of Calabria, Rende, 87036, Italy; email: g.fortino@unical.it; Y. Hao (corresponding author) and L. Hu (corresponding author), Huazhong University of Science and Technology, School of Computer Science and Technology, Wuhan, 430074, China; emails: yixuehao@hust.edu.cn, longhu.cs@gmail.com; I. Humar, University of Ljubljana, Faculty of Electrical Engineering, 1000, Slovenia; email: iztok.humar@fe.uni-lj.si.

1533-5399/2019/04-ART30 $15.00
https://doi.org/10.1145/3239565

Additional Key Words and Phrases: Cognitive computing, cloud computing, edge computing, mobile cloud computing, service migration

ACM Reference format:
Min Chen, Wei Li, Giancarlo Fortino, Yixue Hao, Long Hu, and Iztok Humar. 2019. A Dynamic Service Migration Mechanism in Edge Cognitive Computing. *ACM Trans. Internet Technol.* 19, 2, Article 30 (April 2019), 15 pages.
https://doi.org/10.1145/3239565

1 INTRODUCTION

The massive proliferation of personal computing devices is opening up new human-centered designs that blur the boundaries between humans and machines [1]. Now, the frontier for research on data management is related to the so-called edge computation and communication, consisting of an architecture of one or more collaborative multitude(s) of computing nodes that are positioned among edge networks with the access of cloud-based services. Such a mediating level is responsible for carrying out a substantial amount of data storage and processing to reduce the retrieval time and have more control over the data with respect to cloud-based services, and to consume fewer resources and less energy to reduce the workload [2, 3].

The edge computing paradigm has multiple advantages. First, the edge node can reduce the traffic load of backhaul by providing a certain amount of computing capability, which is significant for applications such as online games that need to transmit 60 or even 120 frames per second. As an alternative solution, the server only sends parameters such as character position, timestamp, and attribute changes (some common data) and leaves the edge node to compute and render the visual image. Second, as a result of the large number of edge nodes deployed in 5G and the big-data analysis based on user preference, the popular contents can be acquired in advance in the interconnecting edge devices, which are only one hop away from the user.

However, edge computing is also faced with many challenges. First, the operation and processing capabilities of an edge device are limited and can fail to meet the demands on real-time service, data optimization, and application intelligence. Second, the intelligence of most typical edge-computing services is only embodied in the artificial intelligence (AI)-enabled data storage and processing on the edge. However, the intelligence is missing from the aspects of behavior feedback, automatic networking, load balance, and data-driven network optimization.

Cognitive computing originates from cognitive scientific theory. Now, it makes machines achieve "brain-like" cognitive intelligence through an interactive cognition loop with machine, cyber space, and humans. Compared to big-data analytics, it possesses the following features: (1) it analyzes the existing data and information in cyber space, to improve the intelligence of the machine; (2) the machine reinterprets and explains the information in the existing cyber space and accordingly generates new information—humans also participate in this process; (3) the machine has the cognition of a human, which provides a more intelligent cognitive service. Its enabling paradigms (e.g., agent-based computing) have been researched and the related concrete applications based on cognitive Internet of Things (IoT) platforms and frameworks have been studied in [4–6].

Nevertheless, the cognitive computing application mainly depends on the machine-learning model trained on the cloud, while the real-time inference requests are made by end edge devices, which so far have been the most common deployment mode of the cognitive service. The existing problem of such a mode is the large latency in the network operation and the service delivery. However, if the cognitive service is deployed on the network edge, the latency of the network response to the user request will be greatly reduced, so research into edge deployment for training and inference machines is rapidly increasing.

Therefore, considering both the advantages and disadvantages of edge computing and cognitive computing, a new computing paradigm called Edge Cognitive Computing (ECC) is proposed, which combines edge computing and cognitive computing. Such a new architecture integrates communication, computation, storage, and application on edge networks; it can achieve data and resource cognition by cognitive computing. Moreover, it can provide personalized services nearby, enabling the network to have a deeper, human-centered cognitive intelligence.

The main contributions of this article are as follows:

- We propose a new ECC architecture that deploys cognitive computing at the edge of the network to provide dynamic and elastic storage and computing services. In addition, the design issues of how to fuse these key technologies of cognitive computing and edge computing are illustrated in detail.
- We propose an ECC-based dynamic cognitive service migration mechanism that considers both the elastic allocation of the cognitive computing services and user mobility, to provide a mobility-aware dynamic service adjustment scheme.
- We develop an ECC-based test platform for dynamic service migration and evaluate it by means of several experiments, with the results showing that the proposed ECC can provide dynamic services according to different user demands.

The remainder of the article is organized as follows. Section 2 introduces the related work. Section 3 presents the proposed architecture for edge cognitive computing and design issues in detail. Section 4 introduces the dynamic cognitive service migration mechanism. Section 5 demonstrates the ECC test platform for dynamic service migration. Finally, Section 6 concludes the article.

2 RELATED WORK

The need for on-demand state-of-the-art services (smart sensing, e-healthcare, smart transportation, etc.) and the latency issues that affect the overall Quality of Service (QoS) for various applications have paved the way to the powerful paradigm of edge computing.

There is a lot of research on edge computing with respect to energy efficiency and latency. The cooperation and interplay among cloud and edge devices can help to reduce energy consumption in addition to maintaining the QoS for various applications. However, a large number of migrations among edge devices and cloud servers leads to congestion in the underlying networks. Hence, to handle this problem, [7] presented an SDN-based edge-cloud interplay to handle the streaming of big data in the industrial IoT environment. Edge computing is expected to support not only the ever-growing number of users and devices but also a diverse set of new applications and services. The work in [1] introduced a system that can pervasively operate in any networking environment and allows for the development of innovative applications by pushing services near to the edge of the network.

In terms of security and privacy in edge computing, there is an increasing realization that edge devices, which are closer to the user, can play an important part in supporting latency- and privacy-sensitive applications [8, 9]. Therefore, the security challenges relate to the protection of device data, such that an unauthorized person cannot access the data, providing secure data sharing between the device and the edge cloud, and safe data storage on the edge cloud [10–12].

However, the above research on edge computing mostly focused on solving the communication problems by leveraging computing and storage, like how to reduce the network load, improve the network efficiency, and reduce the transmission delay. In addition, these works did not consider how to solve the personalization of actual AI applications and how to provide elastic storage and computing services.

Applying cognitive computing in various applications for smart cities has been widely researched; for example, the authors in [13] studied the role of intelligence algorithms such as machine learning and data analytics within the framework of smart-city applications, such as smart transportation, smart parking, and smart environment, to address the challenges of big data. [14] also explored how deep reinforcement learning and its shift toward semisupervision can handle the cognitive side of smart-city services. The work in [15] indicated that the application of deep networks has already been successful in big-data areas, and fog-to-things computing can be the ultimate beneficiary of the approach for attack detection because the massive amount of data produced by IoT devices enables deep models to learn better than shallow algorithms. In summary, most of the current research on cognitive computing has focused on the design of algorithms. However, if cognitive computing wants to be applied and deployed on a large scale, it is necessary not only to solve the problems of how to compute but also to solve what to compute and where to compute, which needs to deploy cognitive computing at the edge of the network.

There is some research on applying cognitive computing to edge computing in [16]. The authors first introduced deep learning for IoTs into the edge-computing environment. They also designed a novel offloading strategy to optimize the performance of IoT deep-learning applications with edge computing since the existing edge nodes have limited processing capability. The works in [17] and [18] proposed a novel deep reinforcement learning approach to solve the resource allocation problems in terms of networking, caching, and computing in edge computing. However, the above research did not apply the cognition for applications to guide network-resource optimization, but only considered the resource allocation using some intelligent algorithms, which cannot provide elastic cognitive computing services.

3 THE PROPOSED ECC ARCHITECTURE AND DESIGN ISSUES

The proposed ECC architecture mainly consists of two components: *the edge network* and *the edge cognition* as shown in Figure 1. The edge network mainly provides the access and resource management of various edge devices. The edge cognition mainly relates to the cognition to edge data, involving service data and network and computing resource data. The edge cognition is mainly composed of two core parts, the data cognitive engine and resource cognitive engine. The interaction between the data cognitive engine and resource cognitive engine is the key design issue, which is also shown at the top of Figure 1.

In the architecture, the data cognitive engine mainly relies on cognitive computing technologies, while the resource cognitive engine mainly uses the related technologies of edge computing. By combining key technologies in cognitive computing (i.e., big-data analysis, machine learning, deep learning) with those in edge computing (i.e., computing offload and migration, mobility management, intelligent proactive caching, resource cooperation management), ECC can better solve the problem of communication bandwidth and delay through the fusion of computing, communication, and storage, thus improving the network intelligence. Below we will introduce the ECC architecture in detail from three aspects: resource cognitive engine, data cognitive engine, and the interaction between them.

3.1 Resource Cognitive Engine

This engine can learn the characteristics of edge cloud computing resources, environmental communication resources, and network resources by cognition, and feed back the integrated resource data to the data cognitive engine in real time. At the same time, it can accept the analysis result of the data cognitive engine and realize the real-time dynamic optimization and allocation of resources. As shown in Figure 1, it mainly includes the resource data pool, network software

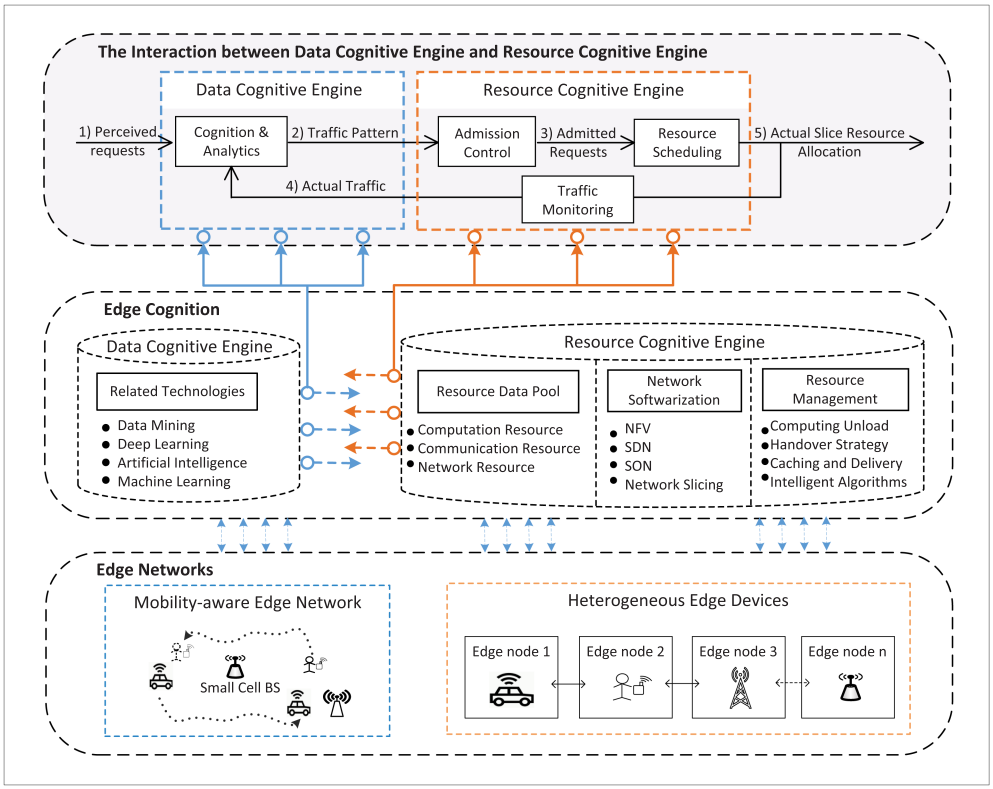

Fig. 1. The edge cognitive computing architecture.

technologies, and resource management technologies. More specifically, the function of this engine includes the following:

(1) **Resource Data Pool:** Realize the massive, heterogeneous, and real-time connection between terminals (such as smart clothing, intelligent robot, intelligent traffic car, and other access devices); ensure the security, reliability, and interoperability of the connection; and constitute the resource data pool (computing resources, communication resources, and network resources) as a basic architecture for data transmission.
(2) **Network Softwarization:** Utilize the network software technologies involving network function virtualization (NFV), software-defined network (SDN), self-organized network (SON), and network slicing to realize high reliability and flexibility, ultra-low latency, and extendibility of the edge cognitive system.
(3) **Resource Management:** Utilize the resource management technologies involving computing unload, handover strategy, caching and delivery, and intelligent algorithms to build a cognitive engine with resource optimization and energy savings to enhance QoE and meet the different demands of various heterogeneous applications.

The key mechanisms involved in the resource cognitive engine include network slicing, computing unloading, caching, and delivery. Using the virtualization technology, network slicing virtualizes the physical infrastructure of 5G into multiple virtualized network slices that are mutually independent and parallel to realize the arrangement and management of the corresponding

network resources. The computing unloading is responsible for the consideration of the computing tasks' assignment problem, aiming to rationally allocate the computing resources on the edge cloud and remote cloud, and thus to complete the computing tasks through cooperation. By caching and delivering, the predicted contents are placed on the edge in advance, and thus the low latency and load reduction of the core network are achieved. SDN/NFV can reduce the deployment costs and improve the efficiency of the network control through the virtualization of the network resources.

3.2 Data Cognitive Engine

This engine deals with the real-time data flow in the network environment, introduces the data analysis and automatic service processing capabilities to the edge network, and realizes cognition of the service data and the resource data by using various cognitive computing methods, including data mining, machine learning, deep learning, and artificial intelligence as shown in Figure 1. The main data sources are:

(1) Collect the external data from the data source in the application environment, such as physical signs and real-time disease risk level under cognitive health surveillance, or real-time behavior information on the mobile user
(2) Collect dynamically the internal data on computing resources, communication resources, and network resources of the edge cloud, such as network type, service data flow, communication quality, and other dynamic environmental parameters

The key point of the intelligent enhancement of the data cognition engine is that multidimensional data (including external data related to the user and the service, and the internal data in the resource network environment) are adopted in cognitive computing technology, which is not the case in the traditional data analysis methods. The data cognitive engine conducts an analysis of the existing data and information (e.g., using the deep convolutional network (DCNN) for facial emotion recognition and using the hidden Markov model (HMM) for user mobility prediction). It then feeds them back to the resource cognitive engine, after which the resource cognitive engine conducts a reinterpretation and analysis of the information to generate new information, which may be further utilized by the data cognitive engine. For instance, in health monitoring, after the monitoring and analysis of the physical health of a smart-cloth-wearing user using cognitive computing, a health-risk level of that user will be obtained; then the resource allocation in the whole edge-computing network will be comprehensively adjusted to the risk level of each user; i.e., the data are utilized for the second time and serve resource allocation and network optimization in turn to form a closed-loop system for cognitive intelligence.

3.3 The Interaction between Data Cognitive Engine and Resource Cognitive Engine

The key design issue of the ECC is the interaction between the data cognitive engine and resource cognitive engine. In edge cognitive computing, we put forward the design idea of realizing the closed-loop optimization with the double cognitive engine to optimize network resource management technology such as a network slice. Here we take cognitive network slicing as an example to illustrate how to fuse the related technologies in cognitive computing and edge computing.

As shown in Figure 1, the data cognitive engine first perceives many requests. The request types of the network-slice service differ from one to another according to different demands (latency, reliability, and flexibility) of different cognitive applications. Then the data cognitive engine will conduct the fusion cognitive analysis of the heterogeneous data based on the current resource distribution situation and real-time requests of the tenant with methods of machine learning and deep learning. Next the data cognitive engine will report the analyzed dynamic traffic pattern to the resource cognitive engine. In the resource cognitive engine, there is a joint optimization

Table 1. Service Resolution for Different Applications

Applications	Service Resolution			Main Metric
	Low	Middle	High	
Emotion Detection	66.3	73.6	79.1	Accuracy (%)
Video Streaming	800 × 600	1280 × 1024	1920 × 1080	Video resolution (pixels)

of the comprehensive benefits and the resource efficiency. First, it conducts admission control to perceived requests, then conducts the dynamic resource scheduling and distribution based on the cognition of network resources, and then feeds the scheduling results back to the data cognitive engine, to realize the cognition of the network-slice resources.

4 DYNAMIC COGNITIVE SERVICE MIGRATION MECHANISM

Under the ECC architecture, due to the mobility of the user, the heterogeneity of the edge device, and the dynamics of the network resources (such as the available storage, the computing resources, and the network bandwidth), we should offer the elastic cognitive service, i.e., offer the service in accordance with the personalized demands of the user. The amount of computation consumed by cognitive computing is particularly large, so the computing resources are required to be more elastic and flexible if deploying the cognitive computing on the edge. The ECC proposed in this article is different from that proposed by those in related work. The ECC mainly focuses on applications related to the artificial intelligence in the IoT, such as automatic pilot, virtual reality, smart clothing, Industry 4.0, emotion recognition, and so forth. In contrast to traditional content retrieval and mobile computing issues, such applications are often more personalized, so the computing resources are required to be more elastic and flexible.

To describe the proposed ECC architecture better, we implemented the Dynamic Cognitive Service Migration Mechanism. Because the device bearing the computing varies, a service migration mechanism is needed. In our ECC-based dynamic service migration mechanism, to reduce the latency, the workload should be finished in the nearest edge device that has enough computation capability at the edge of the network. Thus, according to the user behavior prediction, some contents needed for the service or some jobs for the task are migrated in advance, or the low-resolution work is first migrated to the position to be moved. After the user's pass-by, the service resolution is promoted on that device, thus offering the elastic service.

4.1 Service Resolution

To better explain the elastic service provided by the ECC, we define a new metric called service resolution to evaluate the user QoE. In view of the different applications, the service resolution has different definitions. For example, the emotion detection depends on the accuracy rate and the latency of the emotion recognition, while both are mutually contradictory. A higher accuracy rate needs more computing resources, with higher latency. However, when the user is insensitive to the accuracy rate and pays more attention to the interactive experience, we can provide a low resolution without influencing the user QoE. For the application of video streaming, the service resolution depends more on the resolution of the video streaming acquired by the user. Table 1 lists the service resolution of the two different applications. The emotion detection deems the accuracy rate as a metric, and the video streaming deems the resolution as a metric, respectively offering three services to meet the QoE under different demands of the user, i.e., offering the elastic service and enhancing the user experience.

We will explain how to offer the elastic cognitive computing service from the perspective of the two applications, as follows.

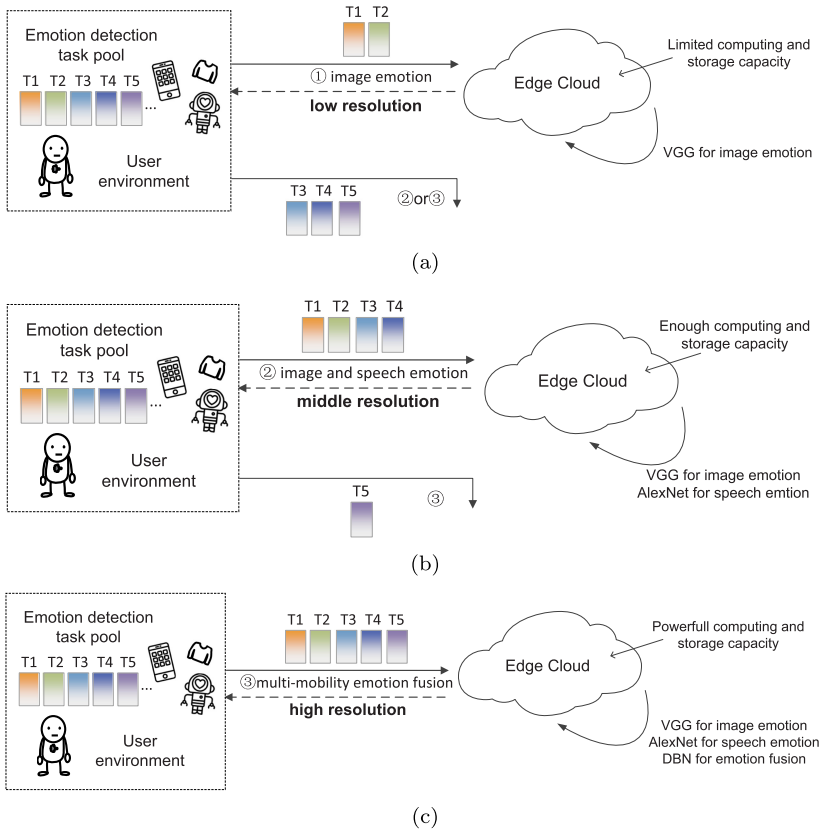

Fig. 2. Service resolution for emotion detection: (a) low resolution, (b) middle resolution, (c) high resolution.

(1) Emotion Detection: As shown in Figure 2, we provide three service resolutions for emotion detection: low resolution, medium resolution, and high resolution. In the case of limited computing resources, we provide low resolution, i.e., only conducting the facial emotion recognition and using the deep neural network VGG. For the medium resolution, we analyze the facial expression (VGG [20]) and speech emotion (AlexNet [19]) simultaneously and carry out the simple decision fusion. For high resolution, we use the strong computing resources, provide the multimodal emotion recognition algorithm, and use the deep network, i.e., deep belief network (DBN), for the decision fusion. For these three service resolutions, the computing resources consumed are increased gradually, and the accuracy rate of the emotion recognition provided is higher.

The user of emotion detection is always a mobile user, so the dynamic change of the mobile computing resources is one of factors influencing the user QoE. In addition, when the user is moving, the network status is changed, but the emotion recognition is required to maintain ultra-high reliability during the communication process. Thus, it is necessary to adopt the elastic computing mode to solve this problem. This application in need of multiple computing decisions was not considered in previous research. It is a mutual contradiction of ensuring the accuracy rate and the latency of emotion recognition at the same time, and a higher accuracy rate needs more computing resources, with higher latency, as shown in Table 2. However, when the user is insensitive to the accuracy rate and

Table 2. Accuracy and Latency of Different Service Resolutions for Emotion Detection

Algorithms	Accuracy (%)	Latency (ms)
VGG	66.3	103.0
AlexNet + VGG	73.6	188.4
AlexNet + VGG + DBN	79.1	265.3

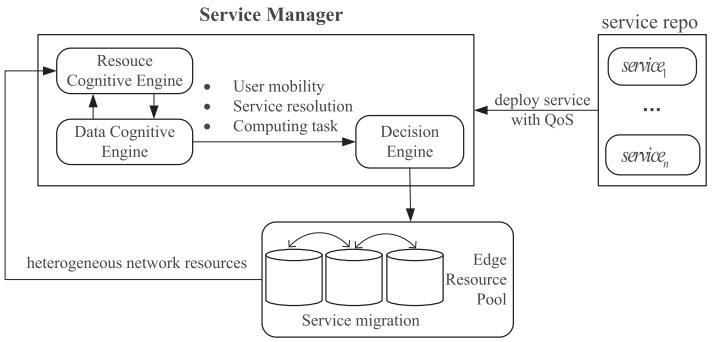

Fig. 3. Service deployment architecture for ECC.

pays more attention to the interactive experience, we can provide low resolution without influencing the user QoE.

(2) Video Streaming: Similar to emotion detection, we provide three service resolutions for a video streaming application, i.e., in consideration of different user demands, user mobility, and a dynamic network environment simultaneously, we provide the video decoding with different resolutions, respectively, and decompose the video decoding task into different resolution tasks in a similar way. When the user is moving, the edge device node better judges whether to conduct the task migration and which resolution of the task migration is conducted according to the user mobility behavior. For example, when the user moves to the other edge node without determining a long-term stay or a short-term stay, the video decoding task with low resolution can be first migrated. In the case of a long-term stay of the user, the high-resolution service can be offered to avoid untimely migration and resource waste. In addition to considering user mobility, migration costs should be considered. The low-resolution service has the lowest migration cost, and the high-resolution service has the highest migration cost.

4.2 Dynamic Service Migration Mechanism

When and how to conduct migration are the two major concerns in dynamic service migration mechanisms. Most migration mechanisms decide when to migrate by only relying on network conditions; few of them take user behavior into account [21, 22]. However, deciding when to migrate according to user behavior and mobility has a large influence on improving user experience and resource utilization.

As shown in Figure 3, the Service Manager implements all the functionalities that an edge node needs to deploy its services. It includes a service repository (service repo) where services ($service_1, \ldots, service_n$) to be provided are stored, e.g., dockerized compressed images or emotion recognition models. The Decision Engine is responsible for deciding which services to

deploy. In Figure 3, the resource cognitive engine manages the computing and network resources of the heterogeneous edge device and cognizes the user mobility, user demands for service resolution, and resource demands for computing tasks in combination with the data cognitive engine. The Decision Engine makes the decision in accordance with the information and migration strategy (based on Q-learning, see below) and accordingly provides dynamic and elastic cognitive services.

The service providers (SPs), i.e., the edge nodes, manage the virtual networks and let $\mathcal{M} = \{1, \ldots, M\}$ be the set of SPs. Let $t \in \{0, 1, 2, \ldots, N\}$ denote the time instant of service request. We assume that the edge device has n services that need to be migrated, and the set of tasks is denoted as $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$. For the migration task T_i, $T_i = \{\omega_i, s_i, o_i\}$, where ω_i is the amount of computing resource required for the task T_i, i.e., the total number of CPU cycles needed to complete the task, and s_i is the data size of the computation task T_i, i.e., the amount of data content to be delivered to the other edge node; specifically, in this work, it stands for the size of the video content or the storage resource consumed by the emotion detection (e.g., the processing code and parameter(s)). Finally, o_i represents the data size of the task result. For instance, in the video decoding case, ω_i is the computing resource needed for the video decoding, s_i is the video data size, and o_i is the data size of the decoded video. After the computation, the Service Provider m sends the transcoded video content back to the user.

Migration Cost: The traffic volume of migrating a virtual server usually cannot be neglected due to the large size of the server states. The migration cost of a virtual server depends on the size of the server as well as the bandwidth available on the migration path. For example, for the emotion detection service, the migration cost depends on the emotion recognition models. For a video-streaming service, the migration cost depends on the data size of the decoded video. A higher service resolution has a higher migration cost.

Migration Goal: Minimize the service costs, and in the meantime, improve the QoE by providing different service resolutions based on user demands, user mobility, and dynamic network resources. For x_t, a service request at time t, represented as (T_i, R, E) where T_i denotes the request task, R denotes the service type and E denotes the expectation of the service request. We define the score (the metric for user-acquired experience) under some migration strategy π as $Score(x_t, \pi)$ and the cost of $Cost(x_t, \pi)$, so the optimization objective can be defined as follows:

$$max\ F(x_t, \pi) = Score(x_t, \pi) - Cost(x_t, \pi), \tag{1}$$

where $Score(x_t, \pi) = \frac{R(x_t) - E(x_t)}{Delay(x_t, \pi)}$, and $R(x_t)$ is the service type acquired by the service request, i.e., the service resolution. We set the value of (0, 1, 2), respectively corresponding to low, medium, and high service resolution. $E(x_t)$ is the expectation of a service request. $Delay(x_t, \pi)$ is the time of service acquisition under strategy π, relevant to ω_i and o_i. $Cost(x_t, \pi)$ is relevant to s_i.

From the definition of $Score(x_t, \pi)$, it is observed that, in the case of a definite latency and a definite service demand of the user, the higher the service resolution provided is, the higher the user experience gained. While providing the same service resolution, the higher the user expectation is, the lower the score is. $R(x_t) - E(x_t)$ can well reflect the relationship between the service acquired by the user and the user expectation. This means that we can provide the low-resolution service if the quality of the service requirements of the user is not high, so as to reduce the energy consumption without influencing the user QoE. When the service acquired by the user and the user expectation are definite, the higher the delay of service acquired is, the lower the score is.

Optimal Problem Formulation: Our problem can be described as a reinforcement learning scenario. The objective is to find an agent that makes the optimal migration policy for each service

request. The optimal migration policy denoted by π^* can maximize the system reward given by

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{x \in \mathcal{X}} F(x, \pi). \tag{2}$$

Let S_i denote the state of environment at time i, defined by the locations of the n services at that time. For a sequence of batch requests $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$, the goal of the service migration is to determine $S_1, S_2, \ldots, S_N$ to maximize the system reward defined by Equation (2).

Q-learning is one of the most popular Reinforcement Learning [23] (RL) methods that is applied in many research areas. The general procedure of the Q-learning algorithm is shown as Algorithm 1.

We define the reward after the action a taken on S_t as

$$R^a_{t+1} = Score(S_{t+1}) - (Score(S_t) + Cost(S_t, S_{t+1})). \tag{3}$$

Similarly, we can also construct a matrix Q to memorize the experience that the agent has gained from the environment. The Q-value of the state-action pair, $Q(S_t, a)$, represents the expected total benefits caused by action a taken in state S_t. The solution is to exploit from the initial state to a final optimal state through updating accordingly by Algorithm 1. In each iteration of the algorithm, the agent observes the current state S and takes action a to move to the next state S' by receiving an immediate reward R_{t+1}, which is used to update the $Q(s, a)$ by following Equation (4), and then begins the next iteration:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)). \tag{4}$$

The α means learning rate, which determines how much the new information overwrites the old. The discount factor γ gives more weight to the most recent reward than others in the future.

ALGORITHM 1: Q-learning algorithm

Initialization $Q(s, a)$
Repeat (for each episode):
  Initial state S
  Repeat (for each step in episode):
    Use some policy such as (ϵ - greedy), and select an action for execution based on the state S
    After executing the action, observe reward and new state S'
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$
    $S \leftarrow S'$
  End

5 TESTBED AND PERFORMANCE EVALUATION

To verify the proposed architecture, an ECC test platform was set up, and a performance evaluation of the dynamic service migration mechanism was conducted in the experimental testbed for the user mobility.

To create the ECC environment, we used several edge-computing nodes that realize the functions of emotion detection and video streaming as shown in Figure 4(a). We also used an Android phone as a user mobile device, designed the Android application program as shown in Figure 4(b), and realized the signal monitoring of the edge-computing node, task uploading, result downloading, and service migration. Figure 4(c) illustrates the software interface running on Windows. Figure 4(d) shows the UI of the emotion detection application.

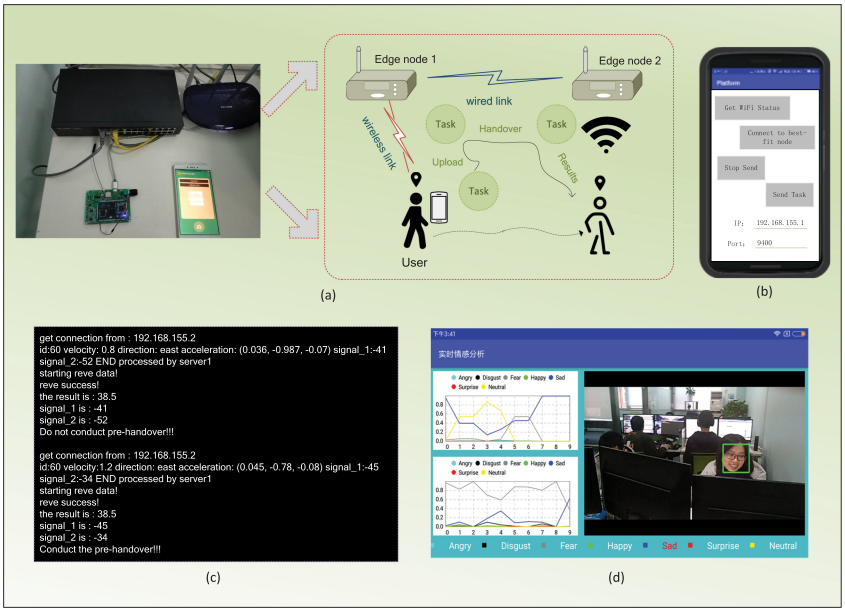

Fig. 4. Experimental platform: (a) hardware platform, (b) interface of mobile application, (c) software interface of the edge node, (d) UI of emotion detection application.

Table 3. Experimental Parameters

Parameter	Value	Description
$B_{i,j}$	5Mbps	The bandwidth between edge node SP_i and SP_j
Q_{T_i}	100Mcycles	The required number of CPU cycles to complete task T_i
O_{T_i}	1Mbits	The content size for task T_i
α	0.01	The learning rate of algorithm
γ	0.8	The discount factor giving more weight to the near future

In the experimental setup, we use four edge nodes and two servers, i.e., $m = 4, n = 2$. For the performance comparison, two schemes are compared with the proposed ECC-based scheme: (1) no migration scheme: service migration is not considered; (2) nearest migration scheme: if needed, service will be migrated to a close access point.

Table 3 lists the values of important parameters considered in the experiments. The task load in the high-resolution migration was 256MB, the task load in the middle-resolution migration was compressed to 128MB, and the task load in the low-resolution migration was compressed to 64MB. The transmission bandwidth between the edge nodes was 5Mbps.

Figures 5 and 6 plot the experimental results for the performance analysis. Figure 5 shows the convergence performance of different scenarios in the proposed scheme using the deep reinforcement learning algorithm. From Figure 5, we can see that the total utility (the cumulative rewards, i.e., the object function F defined in Equation (1)) of the different scenarios in the proposed scheme is very low at the beginning of the learning process. With the increase in the number of episodes, the total utility increases until it reaches a relatively stable value, which is around 400 in the scenario that provides high resolution. We can also observe that the rewards of different resolution services are almost the same at the beginning, and at a stable stage high resolution obtains the

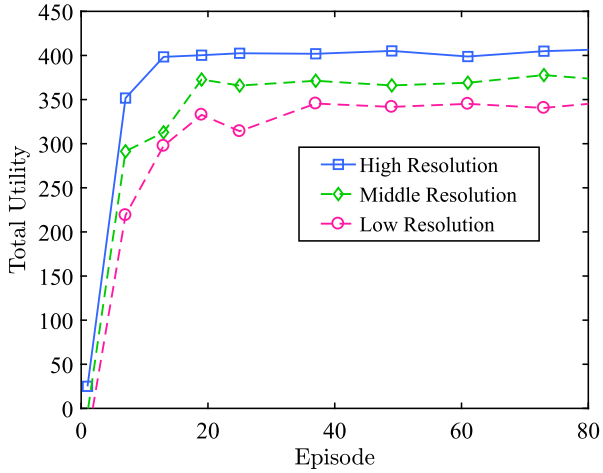

Fig. 5. Convergence performance with difference service resolutions.

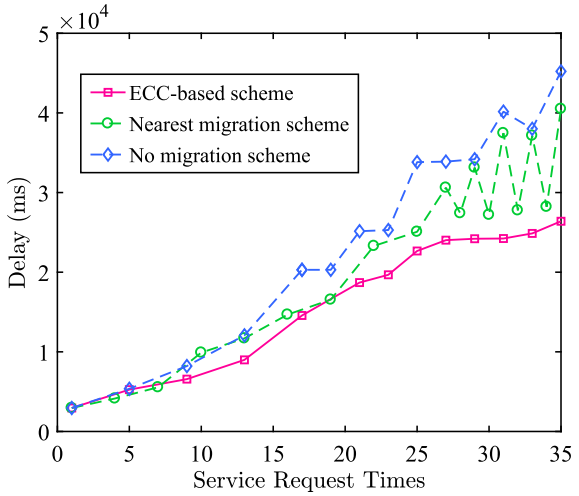

Fig. 6. Delay comparison of different migration schemes with request times.

highest reward, while low resolution obtains the lowest reward. Therefore, the low-resolution service could be first migrated to the corresponding edge node when the user moves to the other edge access point, and then the high-resolution service should be provided at a stable stage.

Figure 6 shows the response time of the edge node providing the emotion detection service with low resolution (the first algorithm introduced in Table 2). In general, the response time is proportional to the number of concurrent requests, which means that the higher the number of concurrent requests is, the longer the average response time is. We compare the time delay under different schemes; from Figure 6, we can obtain that the delay of all the schemes increases with the increase of the service request times, and our proposed RL-based scheme under the ECC architecture has better performance with the lowest latency. This is because these services could be better migrated in advance to the optimal location based on a user-mobility prediction. However, the nearest migration scheme decides to migrate the service just when the user moves to the other access point, which would result in a longer delay and even to lead service disruption. From

Figure 6, we can also observe that due to user mobility, the delay jitter is more severe and the delay difference of these three schemes is bigger when the service request time is higher than 25. The ECC-based scheme is able to reduce the jitter from the results for it learns from the user mobility continuously. Also, it is obvious that the delay was the longest under no migration scheme.

6 CONCLUSIONS

This article presents an ECC network architecture and introduces the key issues. In addition, an ECC platform for dynamic service migration based on a mobile user's behavioral cognition was developed and experimentally tested. The experimental results show that the proposed ECC architecture can simultaneously provide higher QoE compared with the general edge-computing architecture without data and resource cognitive engines that achieve the user behavior prediction to better guide the service migration based on traffic data and the network resource environment. The results effectively demonstrated that edge cognitive computing realizes the cognitive information cycle for human-centered reasonable resource distribution and optimization.

REFERENCES

[1] C. A. Sarros, S. Diamantopoulos, S. Rene, I. Psaras, A. Lertsinsrubtavee, C. Molina-Jimenez, P. Mendes, R. Sofia, A. Sathiaseelan, G. Pavlou, J. Crowcroft, and V. Tsaoussidis. 2018. Connecting the edges: A universal, mobile-centric, and opportunistic communications architecture. *IEEE Communications Magazine* 56, 2 (2018), 136–143.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. 2016. Edge computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.

[3] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab. 2016. Edge computing enabling the Internet of Things. In *Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT'16)*. 603–608. DOI: 10.1109/WF-IoT.2015.7389122

[4] M. Chen, J. Yang , X. Zhu, X. Wang, M. Liu, and J. Song. 2017. Smart home 2.0: Innovative smart home system powered by botanical IoT and emotion detection. *Mobile Networks and Applications* 22 (2017), 1159–1169.

[5] G. Fortino, R. Gravina, W. Russo, and C. Savaglio. 2017. Modeling and simulating internet-of-things systems: A hybrid agent-oriented approach. *Computing in Science and Engineering* 19, 5 (2017), 68–76.

[6] M. Chen, Y. Miao, Y. Hao, and K. Hwang. 2017. Narrow band Internet of things. *IEEE Access* 5 (2017), 20557–20577.

[7] M. Chen and Y. Hao. 2018. Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications* 36, 3 (2018), 587–597.

[8] L. Petri, O. F. Rana, J. Bignell, S. Nepal, and N. Auluck. 2017. Incentivising resource sharing in edge computing applications. In *Economics of Grids, Clouds, Systems, and Services (GECON'17)*, C. Pham, J. Altmann, and J. Bañares (Eds.). Lecture Notes in Computer Science, Vol. 10537. Springer, 204–215.

[9] Y. Qian, M. Chen, J. Chen, M. Hossain, and A. Alamri. 2018. Secure enforcement in cognitive internet of vehicles. *IEEE IoT Journal* 5, 2 (2018), 1242–1250.

[10] M. Villari, M. Fazio, S. Dustdar, O. Rana, L. Chen, and R. Ranjan. 2017. Software defined membrane: Policy-driven edge and internet of things security. *IEEE Cloud Computing* 4, 4 (2017), 92–99.

[11] L. Zhou, D. Wu, Z. Dong, and X. Li. 2017. When collaboration hugs intelligence: Content delivery over ultra-dense networks. *IEEE Communications Magazine* 55, 12 (2017), 91–95.

[12] L. Zhou, D. Wu, J. Chen, and Z. Dong. 2018. Greening the smart cities: Energy-efficient massive content delivery via D2D communications. *IEEE Transactions on Industrial Informatics* 14, 4 (2018), 1626–1634.

[13] H. Habibzadeh, A. Boggio-Dandry, Z. Qin, T. Soyata, B. Kantarci, and H. T. Mouftah. 2018. Soft sensing in smart cities: Handling 3Vs using recommender systems, machine intelligence, and data analytics. *IEEE Communications Magazine* 56, 2 (2018), 78–86.

[14] M. Mohammadi and A. Al-Fuqaha. 2018. Enabling cognitive smart cities using big data and machine learning: Approaches and challenges. *IEEE Communications Magazine* 56, 2 (2018), 94–101.

[15] A. Abeshu and N. Chilamkurti. 2018. Deep learning: The frontier for distributed attack detection in fog-to-things computing. *IEEE Communications Magazine* 56, 2 (2018), 169–175.

[16] M. Chen and V. Leung. 2018. From cloud-based communications to cognition-based communications: A computing perspective. *Computer Communications* 128 (2018), 74–79.

[17] Y. He, N. Zhao, and H. Yin. 2018. Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology* 67, 1 (2018), 44–55.

[18] M. Chen, Y. Hao, M. Qiu, J. Song, D. Wu, and I. Humar. 2016. Mobility-aware caching and computation offloading in 5G ultradense cellular networks. *Sensors* 16, 7 (2016), 974–987.

[19] S. Zhang, S. Zhang, T. Huang, and W. Gao. 2017. Speech emotion recognition using deep convolutional neural network and discriminant temporal pyramid matching. *IEEE Transactions on Multimedia* 20, 6 (2017), 1576–1590.

[20] M. Chen, X. Shi, Y. Zhang, D. Wu, and M. Guizani. 2017. Deep features learning for medical image analysis with convolutional autoencoder neural network. *IEEE Transactions on Big Data* 1 (2017), 1–1. DOI: 10.1109/TBDATA.2017.2717439

[21] A. Machen, S. Wang, K. Leung, B. J. Ko, and T. Salonidis. 2017. Live service migration in mobile edge clouds. *IEEE Wireless Communications* 25, 1 (2017), 140–147.

[22] V. Medina and J. Garcia. 2014. A survey of migration mechanisms of virtual machines. *ACM Computing Surveys* 46, 3 (2014), 30–62.

[23] K. Hwang and M. Chen. 2017. *Big Data Analytics for Cloud/IoT and Cognitive Computing*. Wiley, UK., 2017. ISBN: 9781119247029.

Received December 2017; revised June 2018; accepted July 2018