Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 3057397, 11 pages
http://dx.doi.org/10.1155/2016/3057397

Research Article

Hierarchical Brokering with Feedback Control Framework in Mobile Device-Centric Clouds

Chao-Lieh Chen,[1] Chao-Chun Chen,[2] and Chun-Ting Chen[1]

[1]*Department of Electronic Engineering, National Kaohsiung First University of Science and Technology (First Tech), No. 1 University Road, Yanchao District, Kaohsiung 824, Taiwan*
[2]*Department of Computer Science and Information Engineering and Institute of Manufacturing Information and Systems, National Cheng Kung University, Tainan, Taiwan*

Correspondence should be addressed to Chao-Lieh Chen; frederic@ieee.org

Received 11 April 2016; Accepted 13 June 2016

Academic Editor: Young-June Choi

We propose a hierarchical brokering architecture (HiBA) and Mobile Multicloud Networking (MMCN) feedback control framework for mobile device-centric cloud (MDC2) computing. Exploiting the MMCN framework and RESTful web-based interconnection, each tier broker probes resource state of its federation for control and management. Real-time and seamless services were developed. Case studies including intrafederation energy-aware balancing based on fuzzy feedback control and higher tier load balancing are further demonstrated to show how HiBA with MMCN relieves the embedding of algorithms when developing services. Theoretical performance model and real-world experiments both show that an MDC2 based on HiBA features better quality in terms of resource availability and network latency if it federates devices with enough resources distributed in lower tier hierarchy. The proposed HiBA realizes a development platform for MDC2 computing which is a feasible solution to User-Centric Networks (UCNs).

1. Introduction

A user possesses many mobile devices nearby to enjoy proactively provided services. Wearable and portable devices together with proximity and remote servers process large amount of requests to complete a service. Therefore, an instance of cloud computing methodology to federate these devices and servers is desired especially to the 5th-generation (5G) era that cooperation among massive devices is one of technology focuses. Moreover, the latest surveys [1–5] on User-Centric Network (UCN) advocate self-organizing autonomic networks in which users cooperate through sharing of network services and resources. The self-organizing autonomic architecture, no matter whether ad hoc or infrastructure-based, federates nominal low cost devices and makes users a new type of resource provider and stakeholder in addition to content consumer or producer. Based on sociological relations, mobile devices proactively join the UCN and become network elements (such as router or gateway) to share bandwidth among themselves and to obtain more reliable network connection allowing free roaming. The user-provided networks (UPNs) proposed in [2] are one of the examples. The contribution [3] based on software-defined networking (SDN) explores cooperation among wireless mobile devices to share network transmission bandwidth. Routing, opportunistic relaying, and sensing of hybrid types of information [5] through spontaneous networks [4] are projection of user behaviors in their social networks.

The UCNs emphasize three main key techniques [1]: (1) understanding user context, (2) profiling and predicting user interests, and (3) personalizing content delivery. Therefore, the state observation of UCN underlying a specific service is essential for constructing control and management planes. However, depending on resource distribution among attending mobile devices and data centers, the state space can be too large to be observed. Then, design of control and management algorithms becomes complex and challenging.

In this paper, we propose HiBA architecture with hierarchical brokering and feedback control framework MMCN

for both control and management planes in UCNs. The idea comes from the hierarchical fuzzy control [6] where a complex control problem with huge state space is conquered. In a hierarchical control system, the cross product of state spaces of all control hierarchies remains huge while each hierarchy adopts only a few fuzzy rules to perform simple control tasks.

A series of contributions adopting MDC2 for improving load balance [7], user-centric security [8], access control [9] in the cloud as well as MDC2 applications of real-time seamless video sharing [8], and sociological ontology-based hybrid recommendation [10] were proposed. In these contributions, we see that MDC2 is a feasible solution to provide services in UCN. In this paper, HiBA with feedback control framework generalizes the brokering and state observation. Availability and network latency are analyzed and experimental HiBA comprising real-world mobile devices and servers are realized. We prove that a UCN with huge state space is observable and hence controllable through HiBA cloud networking. In this paper, we further extend our previous presentation [11] to demonstrate algorithm embedding using the MMCN development platform in both lower and higher tiers of HiBA. We study energy-aware balancing in the cloudlet tier where dense mobile devices federate. In addition, we study the load balancing in bigger data centers where large amount of requests usually arrives in a short time. Through both theoretical analysis and real-world experiments, we conclude that a HiBA federation with MMCN is a platform for developing distributed algorithms in future 5G's massive machine-to-machine (M2M) communications and UCNs.

This paper is organized as follows. Section 2 depicts the HiBA architecture and the feedback control framework. Mathematical analysis on availability and network latency is provided in Section 3. Section 4 demonstrates real-world experiments in which the HiBA federates heterogeneous mobile and fixed devices in three tiers using different network interfaces. Energy-aware balancing for a cloudlet and load balancing for a bigger data center are both conducted. We also present conclusions at the end of Section 5.

2. System Architecture

Unlike traditional data centers that span trees of VMs in a top-down manner from a primary computing domain, usually called domain zero, in a physical machine (PM), the proposed HiBA architecture is initiated from physical mobile devices at the bottom tier, called the cloudlet tier. Each broker dynamically connects and adaptively controls lower tier PMs or VMs and thus can avoid drawbacks, such as the degradation of the aggregate available bandwidth and circuitous communication path of VM spanning trees [12]. The system includes resources in the cloudlets then associates cloudlets with private and public clouds to further scale up the cloud federation. On the management aspect, we exploit adaptive feedback control framework to tackle the uncertain dynamics of mobile ad hoc networks in order to adapt to the dynamics such as dynamic joining and leaving a broker's domain caused by user mobility, device failure, or physical network handoff [8].

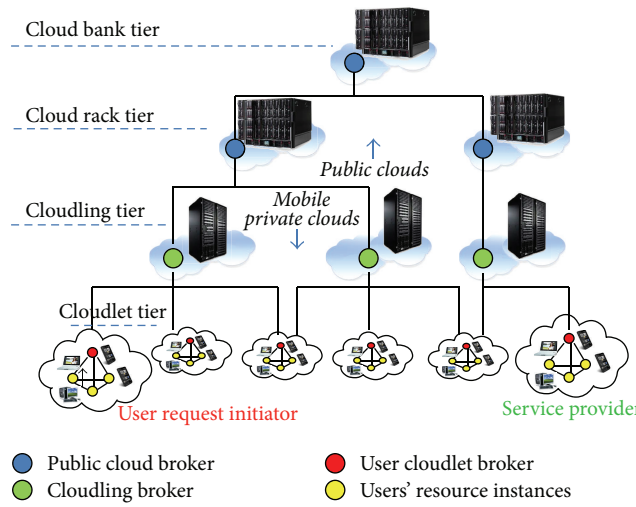


Figure 1: Hierarchical brokering architecture (HiBA) for mobile device-centric cloud computing.

2.1. Hierarchical Brokering. Figure 1 shows the HiBA architecture, which is comprised of four tiers. Tier 0, called the cloudlet tier, contains groups of mobile user devices. Each cloudlet is managed by a cloudlet broker maintaining network connectivity and member association and also performing QoS control. A cloudlet is dynamically organized by the broker according to resource instant status including CPU, memory, network capability, and energy consumption. The resources underlying a tier is abstracted as a JSON or XML list in its broker's database accessible to federation members through RESTful API. Thus, each entry of the list includes a URL containing available RESTful command or a URL of filename and file attributes. Fields of security configurations, NAT port, and GPS coordinates are optionally included in each entry [8]. A tier-0 device does not have a VM member except itself and it aggregates its resources to be shared with other members in the resource list. Each broker hierarchically aggregates the resource lists from members and uploads updates to its upper tier broker.

We continue using the cloudlet federation procedure in our previous research [8]. A cloudlet broker is either dynamically elected among user devices according to the resources status or selected by its upper tier (tier 1) broker, called cloudling broker. A cloudling broker is a small data center or private cloud proximate to a cluster of cloudlets that associate with the cloudling. It is similar to the small data center proposed in [13, 14], owned by a smaller business unit such as a coffee shop or a clinic. We differ from [13, 14] in that our cloudlets are autonomously grouped by user devices which also share resources with others. A tier-n broker associates with a tier-$(n+1)$ broker to scale the federation in depth or alternatively to include more tier-$(n-1)$ federations to scale in width. In the proposed architecture, each broker only recognizes next lower tier brokers while further lower tiers are transparent to it. Furthermore, each broker has the same feedback control framework for managing lower tier devices' network attachments and resource associations as well as for QoS control on requesting and executing services.

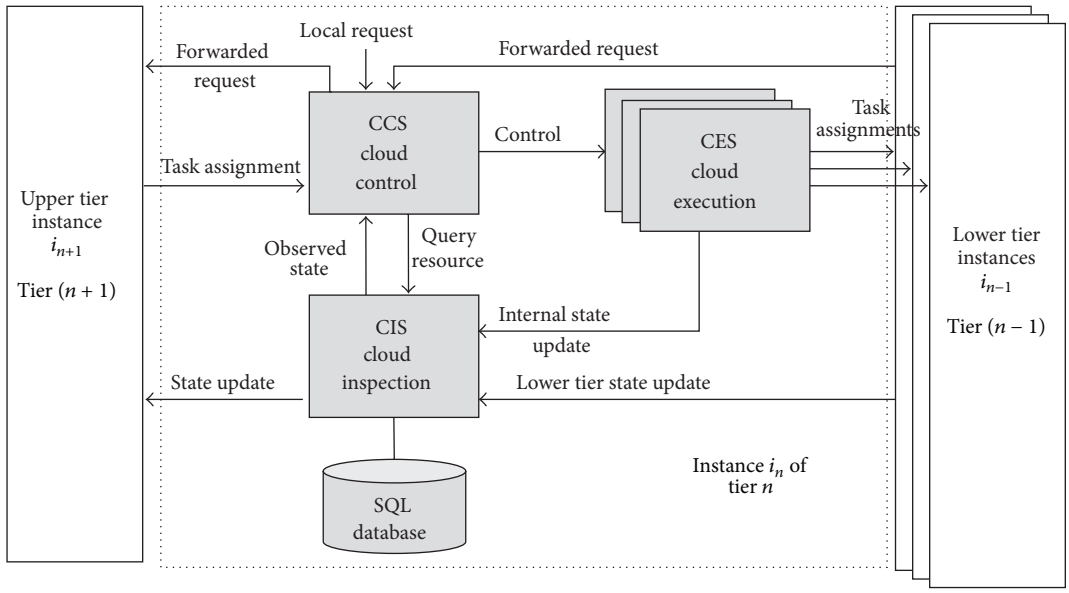


Figure 2: The feedback control framework MMCN in a HiBA broker.

2.2. Feedback Control Framework. Figure 2 shows the feedback control framework MMCN of each broker in the HiBA virtualized network. The feedback loop is used to adaptively manage and control the cloud federation governed by the broker. This management includes network attachments and resource associations caused by user mobility and VM migration [8]. In future application, the framework is ready for developing hierarchical control algorithms such as request differentiation and task scheduling as well as real-time QoS control and job tracking once a service is started. A feedback control loop comprises three subsystems including Cloud Inspection Subsystem (CIS), Cloud Control Subsystem (CCS), and Cloud Execution Subsystem (CES) which are analogous to observer, controller, and plant, respectively, in a classic feedback control system. When performing real-time control for QoS, multiple lower tier instances of the feedback loop are allocated. A tier-n CCS determines the number of tier-$(n-1)$ instances to be allocated; then the tier-n CES invokes the required instances each of which is constituted by the other set of CIS, CCS, and CES at the tier-$(n-1)$.

A CCS processes tasks assigned by its upper tier broker and processes requests from lower tiers. Then according to the database and states tracked by the CIS subsystem, it determines the controls including admission of member joining, allocating member federations for task execution, schedules of tasks, and service level agreements (SLAs) of tasks to be assigned to member federations. A number of CES instances physically accomplish the network attachment for the admission control as well as the recruiting of members associated with corresponding SLAs to complete current schedule. A CCS produces these determinations as controls to CES subsystems while the executions are left to CESs and associated member federations. The CIS observes the state updates from lower tiers for CCS's reference when determining these controls.

2.3. Algorithm Embedding. It is obvious that this framework is feasible for future development of hierarchical control and management algorithms including admission control, request differentiation, task partition, adaptive resource allocation, and dynamic task scheduling regarding SLAs assigned by upper tier broker. For example, on receiving a new request, a priority queue of requests is adapted with new set of priorities. The CCS checks with the CIS to determine whether to forward the request to upper tier broker or to reduce the request into smaller tasks such that the CESs can further assign the task partitions to lower tier members just as the processing of task assignments from upper tier broker's CES. The CES's makes lower tiers transparent to the CCS and upper tiers as if it is the single plant of the CCS controller. Since it is the CESs and associated member federations executing tasks rather than the CCS, the CCS is able to process the management and control algorithms without waiting for the end of the current job execution. This is easy to be realized by programming multiple threads each of which realizes CCS, CESs, and CIS subsystems.

In summary, the CCS of a tier-n broker forwards requests to tier-$(n+1)$ if it is not able to process them according to the CIS database. The tier-$(n+1)$ broker assigns jobs to other tier-n brokers through CES if the requested resources are sharable in these tier-n members and below. Therefore, the sharing becomes intercloud service of tier-n as well as intracloud service of tier-$(n+1)$.

3. Performance Analysis

The baseline performance for benchmarking is to evaluate the HiBA architecture and feedback control framework without optimization on specific management and control. That is, the queues are simple FIFOs without priority adaptation and all devices in all tiers randomly generate requests. The

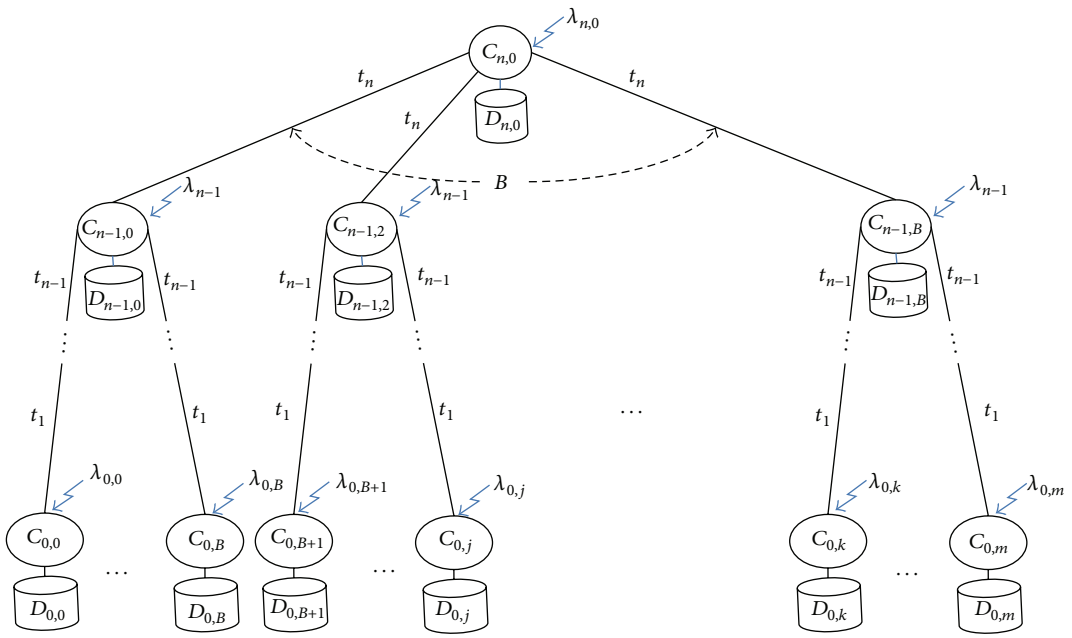


Figure 3: Performance model of the HiBA architecture.

performance model of the virtualized network is shown in Figure 3. Suppose that the maximum number of members of a HiBA broker is B; a broker j in tier i has computing capability C_{ij}, which is also the minimum length of the *Task Queue*. The capability C_{ij} also represents the maximum number of tasks that can be processed by a broker in tier i under the condition that no task is dropped. The resources are assumed to be uniformly distributed in each tier. In tier i, the average amount of contents and resources is D_{ij}. A tier-i broker has its amount of local requests in a Poisson distribution with mean λ_{ij}. This simulates the reinitiated requests caused by request partitions and task handoffs. t_n denotes the mean initial latency caused by network delay to forward requests and the waiting time that requests stay in the *Request Queue* until being partitioned into smaller tasks. In this paper, we derive baseline performances on availability and latency.

3.1. Resource Availability. We define availability, a_{ij}, as the ratio of computing capability C_{ij} over the number of accepted requests R_{ij} to a broker j in tier i. That is,

$$a_{ij} = \min\left(\frac{C_{ij}}{R_{ij}}, 1\right). \tag{1}$$

In a HiBA architecture of n tiers, the expected total resource amount summing from the records in the CIS databases is approximated as

$$D_{\text{total}} = D_n + \sum_{i=1}^{n}\left(\prod_{k=0}^{i-1} B_{n-k}\right) D_{n-i} = \sum_{i=0}^{n} B^i D_{n-i} \tag{2}$$

supposing that $D_i = E[D_{ij}]$. Thus, the expected total resource amount of the subtree rooted at a tier-i node is

$$D_{i,\text{total}} = \sum_{k=0}^{i} B^k D_{n-k}. \tag{3}$$

For any request, the probability that the requested resource r_{ij} is available in local node ij is approximated by

$$P\left(r_{ij}\right) = P\left\{r_{ij} \in \mathbf{D}_{ij}\right\} = \frac{D_{ij}}{D_{\text{total}}}, \tag{4}$$

where $\mathbf{D}_{ij}$ is the set of resources in node ij. Suppose that the sample space is the union of $\mathbf{D}_{ij}$ in the whole HiBA architecture. Then, the worst case of availability occurs when network latency is much smaller than the mean time interval $1/\lambda_{ij}$ between two consequent requests. That is, requests from other cloud federations including those in lower and higher tiers arrive in current tier instantly with negligible network delay. Thus, the maximal number of requests arriving node ij is

$$R_{ij} = P\left(r_{ij}\right) \sum_{k=0}^{n} B^k \lambda_{n-k,j}. \tag{5}$$

Then we obtain the worst case availability as

$$\begin{aligned} a_{ij} &= \min\left(\frac{C_{ij}}{P\left(r_{ij}\right)\sum_{k=0}^{n} B^k \lambda_{n-k,j}}, 1\right) \\ &= \min\left(\frac{C_{ij}}{D_{ij}} \cdot \frac{D_{\text{total}}}{\sum_{k=0}^{n} B^k \lambda_{n-k,j}}, 1\right). \end{aligned} \tag{6}$$

From (6), we see that when n is smaller, resources (including contents) are more centralized with larger D_{ij} and if D_{total} remains large, this causes lower availability. Thus there are various means to increase availability. One is to increase capability C_{ij}, though this will increase costs. Alternatively, branch amount B of each tier can be increased, though this also will increase computing capability and costs. Third, by adopting HiBA, we put user devices in tier 0 or tier 1 (once the user device is elected cloudlet broker). However, the capabilities C_{0j} and C_{1j} of thin devices may be small, and so D_{0j} and D_{1j} are also expected to be small. This indicates that availability remains close to 1 if resource distribution D_{ij} is proportional to the capability C_{ij} with the ratio of total number of requests over total resource amount. Therefore, the optimal resource distribution to low tier brokers and mobile devices both offloads data centers' computing overhead and increases user satisfaction.

3.2. Latency. Social networks can cause locality, such that many requests do not travel to their destinations over long distances in terms of either network relay counts or the terrestrial radio barriers. Thus initial latency of a service is reduced. Lower tier brokers have smaller resource granules, while locality based on social network provides access proximity and thus results in shorter latency. We estimate the latency of a request traversing the HiBA architecture as follows. Suppose that a request initiated from a tier-i node arrives at its destination possessing the required resources in tier $l, i < l$. The expected latency of the request is

$$T_{i,l} = P\left(\bar{r}_{i<l}\right) \sum_{k=i}^{l} \left(t_{N,k} + t_{D,k} + t_{C,k}\right), \tag{7}$$

where $P(\bar{r}_{i<l})$ is the probability that the requested resource is *not* found in tiers lower than l and $t_{N,k}, t_{D,k}, t_{C,k}$ are latencies caused, respectively, by network communication, request queuing plus database querying, and computing for the brokering at each tier-k broker on the path to tier l. Supposing that request arrival is independent of resource distribution, we have

$$P\left(\bar{r}_{i<l}\right) = \frac{\lambda_i}{\lambda_{\text{total}}} \left(1 - \frac{D_{l-1,\text{total}}}{D_{\text{total}}}\right). \tag{8}$$

Thus, the estimated latency is

$$T_{i,l} = \frac{\lambda_i}{\lambda_{\text{total}}} \left(1 - \frac{D_{l-1,\text{total}}}{D_{\text{total}}}\right) \left(\sum_{k=i}^{l} t_{N,k} + t_{D,k} + t_{C,k}\right). \tag{9}$$

If a request is originated from a lower tier user device, that is, $i = 0$ or 1, the effective way to reduce expected latency $T_{0,l}$ or $T_{1,l}$ is to reduce $P(\bar{r}_{i<l})$. This means increasing $D_{l-1,\text{total}}$ by distributing resources to lower tiers, that is, increasing D_k for $k < l$. However, this also increases both database searching time $t_{D,k}$ and computation time $t_{C,k}$. Considering network delay $t_{N,k}$, we can expect that, in the virtualized network, the physical distance of a cloud server is increasing as k is getting larger. A high tier link in the virtualized HiBA network actually contains many more physical relaying hops than a lower tier link. If request is sent using TCP protocol, the network latency is increasing much more than that caused by database searching and brokerage computing, as k increases. From (9), we still effectively decrease the latency by increasing $D_{l-1,\text{total}}$. This can be expected especially for multimedia content sharing since, according to social network relations, contents are stored in proximate cloudlets or cloudlings which are lower tiers (small l).

3.3. Development Platform for MDC2 Control and Management. From the availability and latency analysis, we see that a HiBA with feedback framework is a development platform. It is easy to observe the performance when developing algorithms for admission control, request differentiation, task partition, adaptive resource allocation, and dynamic task scheduling regarding SLAs assigned by upper tier broker. For example, the admission control in federating a tier affects resource distribution D_{ij} and according to the number of network hops between a broker and a member, the network latency $t_{N,k}$ in (9) also differs. The request differentiation and queuing mechanism directly affect $T_{i,l}$ since $t_{D,k}$ includes the queue delay that a request stays in the *Request Queue*. Task partitioning, resource allocation, and scheduling further affect the efficiency processing requests and consequently they further affect $t_{D,k}$ and resultant $T_{i,l}$. Globally, they also affect arrival rate λ_i at tier i. Exploiting hierarchical feedback control, it is easy to ensure availability by performing proper management while tracking the latency by tuning control regarding the deadline specified in the SLA of each task.

4. Case Studies

To demonstrate algorithm embedding using the MMCN development platform in both lower and higher tiers, we study energy-aware balancing among mobile devices in the cloudlet tier as well as the load balancing in bigger data centers where large amount of requests would arrive in a short time, that is, large λ_{ij}. The energy-aware balancing is critical in crowd-sensing applications especially when the sensing data amount is large such as using cameras for image data collecting. The load balancing is essential for bigger data centers especially when database access frequency is high. Both of the balancing algorithms are effective in the Industry 4.0 era when crowd-sensing and big data analytics are deployed.

4.1. Energy-Aware Balancing. The energy-aware balancing is based on unsupervised fuzzy feedback control where the reference command is also adapted according to the feedback state of the federation self-organized by mobile devices. The principal idea comes from the energy proportional routing [15] that the lifetime of a clustering-based sensor network is prolonged if member nodes' proportions of consumed energy in the remaining are close to the cluster's. We exploit the energy proportion of the cloudlet federation as the adaptive reference to be tracked by the fuzzy feedback control system. Therefore, the energy sharing is unsupervised because of no given objective of the control a priori.

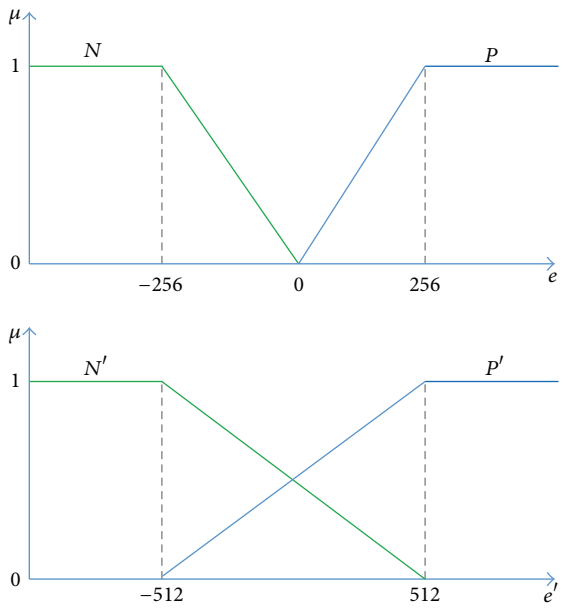


Figure 4: Membership functions for fuzzy sets N, P, N', and P'.

Suppose that, in a cloudlet tier of N member nodes, data transmission is observed by the cloudlet broker in each round t (discrete time). We first define symbols as follows:

(i) k: member node identification, $1 \le k \le N$.

(ii) $D_k(t)$: data transmission amount being assigned to node k at round t.

(iii) $E_k(t)$: remaining energy of node k at t.

(iv) $X_k(t) = D_k(t)/E_k(t)$, representing the ratio of overhead to the remaining energy (current capability). This is the indication of how node k is loaded with respect to the remaining energy.

(v) $Th(t) = \sum_{k=1}^{N} D_k(t) / \sum_{k=1}^{N} E_k(t)$, representing how the whole cloudlet federation is loaded.

(vi) $e_k(t) = X_k(t) - Th(t)$, representing the difference of loading between node k and the whole federation.

(vii) $e_k'(t) = e_k(t) - e_k(t-1)$ representing how the difference changes.

(viii) $r_k(t)$: the ratio of data amount assigned to node k at t.

We define fuzzy rules as follows:

(R1) If $e_k(t)$ is P and $e_k'(t)$ is P', then $r_k(t+1)$ is $S_k(t)$.

(R2) If $e_k(t)$ is P and $e_k'(t)$ is N', then $r_k(t+1)$ is $M_k(t)$.

(R3) If $e_k(t)$ is N and $e_k'(t)$ is P', then $r_k(t+1)$ is $M_k(t)$.

(R4) If $e_k(t)$ is N and $e_k'(t)$ is N', then $r_k(t+1)$ is $L_k(t)$.

The membership functions for fuzzy sets N, P, N', and P' are configured in Figure 4.

We realize the "and" operator with t-norm "minimum." That is, the matching degrees μ_1, μ_2, μ_3, and μ_4 of premise part of rules (R1) to (R4), respectively, are

$$\begin{aligned}
\mu_1(t) &= \min\left(\mu_P(e(t)), \mu_{P'}\left(e'(t)\right)\right),\\
\mu_2(t) &= \min\left(\mu_P(e(t)), \mu_{N'}\left(e'(t)\right)\right),\\
\mu_3(t) &= \min\left(\mu_N(e(t)), \mu_{P'}\left(e'(t)\right)\right),\\
\mu_4(t) &= \min\left(\mu_N(e(t)), \mu_{N'}\left(e'(t)\right)\right).
\end{aligned} \tag{10}$$

Obtaining the matching degrees μ_1, μ_2, μ_3, and μ_4 of the four fuzzy rules, respectively, we perform the defuzzification equivalently applying the Takagi-Sugeno inference method. The inference result, adequate ratio of data amount to transmit, is

$$\begin{aligned}
&r_k(t+1)\\
&= \frac{\mu_1(t) S(t) + \mu_2(t) M(t) + \mu_3(t) M(t) + \mu_4(t) L(t)}{\mu_1(t) + \mu_2(t) + \mu_3(t) + \mu_4(t)}
\end{aligned} \tag{11}$$

by configuring the conclusion part membership functions as dynamic singletons as follows:

$$\begin{aligned}
L(t) &= r_k(t-1) + \Delta\\
M(t) &= r_k(t-1)\\
S(t) &= r_k(t-1) - \Delta,
\end{aligned} \tag{12}$$

where Δ is the ratio tuning amount for each round. Finally, the data amount assigned to node k is determined as

$$D_k(t+1) = \frac{r_k(t)}{\sum_{i=1}^{N} r_i(t)} \mathbf{D}(t), \tag{13}$$

where $\mathbf{D}(t)$ is the total data amount to be transmitted from this cloudlet at time t. In the above fuzzy inference algorithm, each member node tracks the dynamic control goal $Th(t)$ and the proportion of energy to be consumed in the remaining energy approaches the proportion regarding the whole federation. Therefore, the intrafederation energy sharing is achieved by offloading transmission jobs using the proposed algorithm. When each node k is a lower tier federation, the offloading is hierarchically extended through the MMCN networking where the data amount $D_k(t)$ is determined and assigned by CCS and the energy status updates $E_k(t)$'s are received and aggregated by CIS. This shows the scalability of the HiBA architecture.

4.2. Load Balancing. The load balancing is required in a bigger federation organized in a higher tier of HiBA because a higher federation always has larger amount of request arrivals. As shown in Figure 5, the framework of the proposed load-balanced cloud service interface consists of three components which realizes CES, CCS, and CIS, respectively. The details of components are presented as follows:

(1) Cloud Service Interface Node (CSIN): it is a virtual machine that provides cloud service interface and is

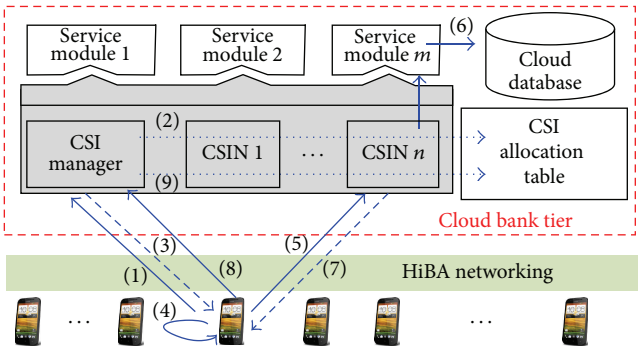


FIGURE 5: Framework of load-balanced cloud service interface.

denoted as $CSIN_i$ in Figure 5. A CSIN can receive members' requests and obtain results by interacting with requested cloud service module. In our design, each cloud service interface in CSIN is implemented as a RESTful API for universally communicating with different types of mobile devices. A CSIN is then the realization of CES in the MMCN framework.

(2) CSI manager: it is designed to manage the usage status of CSINs and matches mobile users to CSINs. The efficiency of the load balance depends on the matching method of the CSI Manager. A CSI manager is then the realization of CCS in the MMCN framework.

(3) CSI allocation table: it maintains the serving status between mobile users and CSINs and is used by the CSI manager for assisting the matching decision for the new-arrived mobile users. This table is stored in the cloud database. A CSI allocation table is then the realization of CIS in the MMCN framework.

The processing flow of using a cloud service for a mobile user is designed for achieving the load balance. The idea of committing requests requires three phases: (1) service registration (Steps (1)-(2)), (2) service execution (Steps (3)–(7)), and (3) service deregistration (Steps (8)-(9)). The detailed steps, also illustrated in Figure 5, are presented as follows:

(1) Mobile user MU_i sends a request to CSI manager for allocating a CSI address.

(2) CSI manager searches for a CSI machine with least number of serving users, say, $CSIN_n$, in the CSI allocation table, and then registers (MU_i, $CSIN_n$) into the CSI allocation table.

(3) CSI manager informs MU_i that $CSIN_n$ can serve his/her cloud service requests in the following period.

(4) MU_i configures the cloud service interface by replacing the IP attribute in the service template with the IP address of $CSIN_n$.

(5) MU_i connects to $CSIN_n$ through the configured URL of the cloud service interface and sends a request.

(6) $CSIN_n$ delivers the request to the associated cloud service which may access the cloud databases if required. After processing, the results will be sent back to $CSIN_n$.

(7) $CSIN_n$ sends the results to MU_i. (If more requests are required to processed, Steps (3)–(7) are repeated.)

(8) MU_i deregisters $CSIN_n$ to CSI manager.

(9) CSI manager removes the registration record of MU_i from the CSI allocation table.

Notice that the matching rule for a new mobile user and CSINs in the current design uses least-user-first basis, as shown in Step (2). The matching principle can be modified according to different demands. In the experiments, we will show that the current design already obtains splendid performance. More study on customizing matching methods is one of our future research directions. That is, although five machines are used in our CSI (more than the single-machine CSI), the improvement is by a factor of one hundred, indicating the superior performance of our proposed load-balanced CSI mechanism under the HiBA architecture.

5. Real-World Experiments

5.1. HiBA Baseline Performance. We conduct an experiment demonstrating performance difference in terms of availability and latency when distributions of virtualized resource instances (VRIs) differ. We leave development of enhancing algorithms for specific purposes in the CCS as future work which will seek performance in terms of various metrics. A total of 50,000 VRIs are distributed in a 3-tier HiBA federation comprising Android smart phones, tablets, and VMs in desktop PCs. Figure 6 is the HiBA topology and related specifications of the machines are shown in Table 1. Each device is labeled (i, j) if it is the jth node at tier i. Device (1, 3) connects to (2, 0) through a 3G access point with VPN tunneling. Device (0, 11) connects to (1, 0) with USB and shares Internet connection from (1, 0). These machines have different computing ability, though request and task queues being of the same size of 20 to preserve the differences of computing ability. Requests are randomly generated in each device. Mean request arrivals (λ) are 3, 1.5, and 1 which equivalently mean that request intervals ($1/\lambda$) are 333 ms, 666 ms, and 1 s, respectively. The resource distributions have four cases. The first case is that the tier-2 device possess all the 50,000 VRIs while 16,000 VRIs are duplicated to each tier-1 federations. In the remaining three cases, we, respectively, duplicate 28000, 32000, and 36000 VRIs to each tier-1 federation. The experiment is conducted in a heterogeneous networking environment connecting machines by different communication technologies shown in Table 1.

The results of the experiment are shown in Figures 7 and 8. Both Figures 7 and 8 reveal that distributing resources in lower tiers provides better performance. Figures 8(a) and 8(b) show the latency differences caused by the resource distributions that 16000, 32000, and 40000 VRIs are duplicated in each cloudling tier. The results before all brokers fully loaded are also given with respective zoom-in charts. When devices all continuously issue requests at a high frequency (3 requests per second), the queue delays are obviously high and we see a few requests that are dropped. Requests from tier-1 devices have lower loss rate since the resources requested are

Table 1: Machines in the offloading experiments using heterogeneous communication technologies.

(Tier, device)	Specifications	Physical network link
Tier 2/(2, 0)	Desktop PC: Intel Core i5-3470 CPU 3.2 GHz, 4 G RAM	Ethernet
Tier 1/(1, 0), (1, 1)	Desktop PC: Intel Core i7-4770 CPU 3.4 GHz, 8 G RAM	
Tier 1/(1, 2)		
Tier 1/(1, 3)	Laptop: Intel Core i7-2820QM CPU 2.3 GHz, 8 G RAM	Down: WiFi; UP: 3G VPN
Tier 0/(0, 0~3)	Desktop PC: Intel Pentium D CPU 3.4 GHz, 1 G RAM	Ethernet
Tier 0/(0, 4~6)		
Tier 0/(0, 7)		
Tier 0/(0, 8)	Tablet: ASUS Fonepad Intel Atom Z2420, 1.2 GHz	WiFi
Tier 0/(0, 9)	Cell Phone S3: SAMUNG Exynos 4412, 1.4 GHz	
Tier 0/(0, 10)	Tablet: ASUS Nexus7 NVIDIA Tegra 3 4-core	
Tier 0/(0, 11)	Tablet: WS-170 Qualcomm QCT MSM8x60 surf/1.5 GHz duo-core	USB ←→ PC ←→ Ethernet
Tier 0/(0, 12~13)	Virtual Machines (VirtualBox)	Ethernet
Tier 0/(0, 14)	Tablet: Samsung Galaxy CPU GT-I8260	WiFi

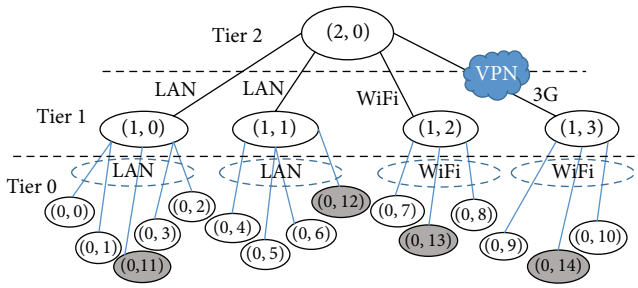


Figure 6: Real-world HiBA topology for offloading performance measurement. Gray nodes are local VMs of respective tier-1 brokers. Diverse physical network links are exploited in tiers of federations.

obtained in a shorter time. For each physical machine, the CPU utilization of the MMCN brokering itself is smaller than 2% depending on the number of cores. The mean brokering computation and database querying time is estimated to be about 500 ms while the WLAN delay time is smaller than 5 ms. In the heterogeneous network case, the latency difference will be larger if upper tiers are physically at a long distance from the cloudlings and cloudlets. If the request interval is higher than the processing time, the latency tends to be smaller. When more VRIs are duplicated in lower tier devices, we also see that the latencies are smaller.

In this paper, we leave algorithms, such as request partition, network embedding, and federation for specific performance metrics, as future work. Instead, we implement real-world HiBA to prove that, with feedback framework, it is a new design paradigm and development platform for these algorithms.

5.2. Energy-Aware Balancing. In the energy-aware balancing experiment we exploit four mobile devices of different types from different manufacturers. The cloudlet is self-organized according to the broker election protocol in [8]. The elected broker is ASUS-Fonepad. The other hierarchical federations are in Figure 6. Each device, including the broker itself, in each round updates its energy status to the CIS of the broker (ASUS-Fonepad). Each mobile device in the cloudlet is recharged to 100% of respective battery energy capacity. The data and total data amount to be transmitted in each round are randomly generated with mean of 2.4 GB prior to the experiment. For each round t, total data amount is the same for both balancing and nonbalancing cases. In nonbalancing case, the data amounts are evenly assigned to members. In balancing case, the data amounts are determined by the fuzzy controller in Section 4.1. The data amount assignment is of unit 100 MB. When any device has remaining energy lower or equal to 40%, the experiment is terminated. The initial singletons are configured as $S(0) = 3$, $M(0) = 5$, and $L(0) = 7$ ($\Delta = 2$). The result of energy-aware balancing is shown in Figure 9. The respective control systems' behaviors are in Figure 10. The fuzzy controllers of mobile devices all track the federation's data amount to energy proportion (Th). We see that no matter how the performance of respective device varies, a device with better transmission capability will share its energy with others. Tablet PCs are usually with higher capacity batteries. Without energy sharing, the remaining energy of a tablet drops more slowly than a mobile phone. With energy sharing, tablets undertake more transmission jobs and become with higher energy drop rate, and the mobile phones become with lower energy drop rate. However, we see that the whole cloudlet federation has longer lifetime. The experiment can also be applied to energy consuming tasks in addition to data transmission.

5.3. Load Balancing. The load-balanced cloud service interface (CSI) in the form of RESTful APIs is implemented by using Jersey and is deployed on Microsoft Windows Azure public cloud platform. We conduct an experiment for comparing the proposed CSI to that of single machine. In our experiment, five CSI machines are used to share the requests from users. In the experiment, the data producing rate for each user is 1 request per second. The performance

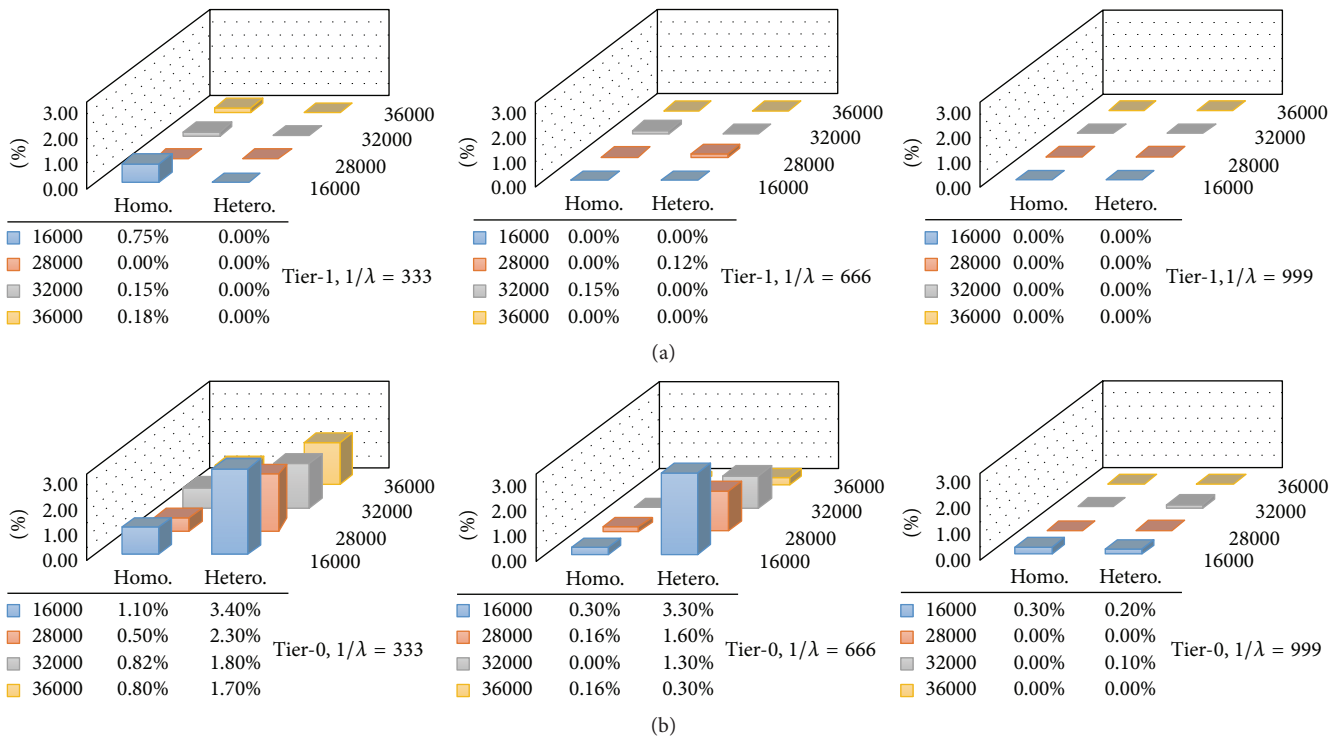
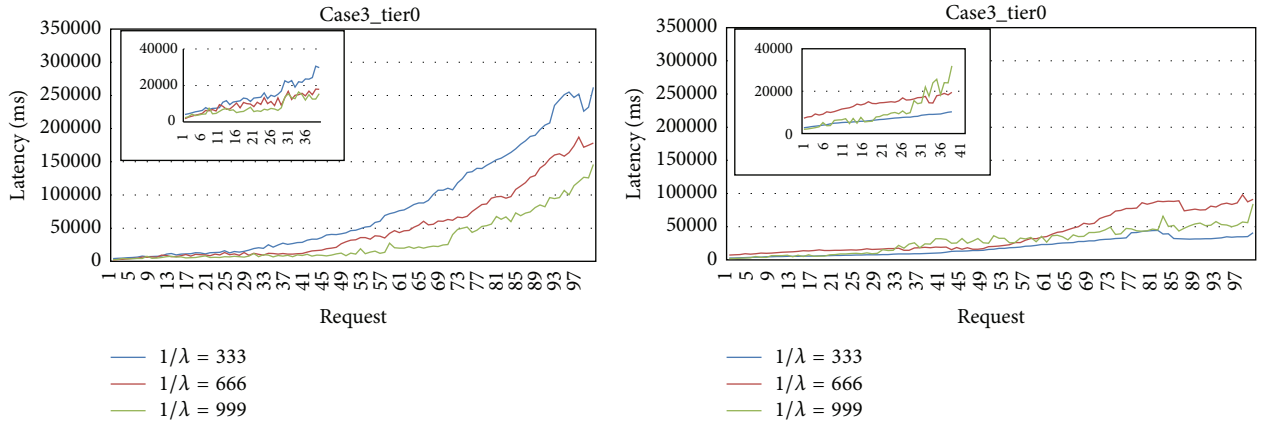


Figure 7: Mean losses of tier 0 (b) and tier 1 (a) in cases of different resource distributions when 16000, 28000, 32,000, and 36,000 VRIs are duplicated to each tier-1 federation. Request intervals $1/\lambda$ are in milliseconds.



(a) 16,000 VRIs in each tier-1 federation

(b) 40,000 VRIs in each tier-1 federation

Figure 8: Mean latencies in cases of different resource distributions.

metric is the missing rate defined as $(T - S)/T$, where T is the number of total requests and S is the number of successful requests. Table 2 shows the experimental results under different amount of users from 100 to 300. We can see that when the number of users is low (100 users), both CSI designs can successfully process their data requests. As the number of users increases (e.g., 300 users), our proposed CSI can almost process it with only 0 : 27% missing rate; however, over 27% requests are failed to deliver data to cloud database in the CSI of single machine.

6. Conclusion

We have proposed and implemented a mobile device-centric cloud computing architecture, HiBA, based on feedback control framework. The proposed hierarchical brokering architecture is self-organized featuring scalability and hierarchical autonomy and is easy to embed management and control algorithms to develop a federation with better performance in terms of availability and latency. Mobile devices and small servers federate into cloudlets and cloudlings of hierarchical

Table 2: Experimental comparison of different CSI designs.

Users	Single			Load-balanced scheme		
	Total requests	Successful requests	Missing requests	Total requests	Successful requests	Missing requests
100	52395	52395	**0%**	53370	53370	**0%**
200	97457	80947	**16.91%**	104244	104244	**0%**
300	113624	81883	**27.94%**	131742	131388	**0.27%**

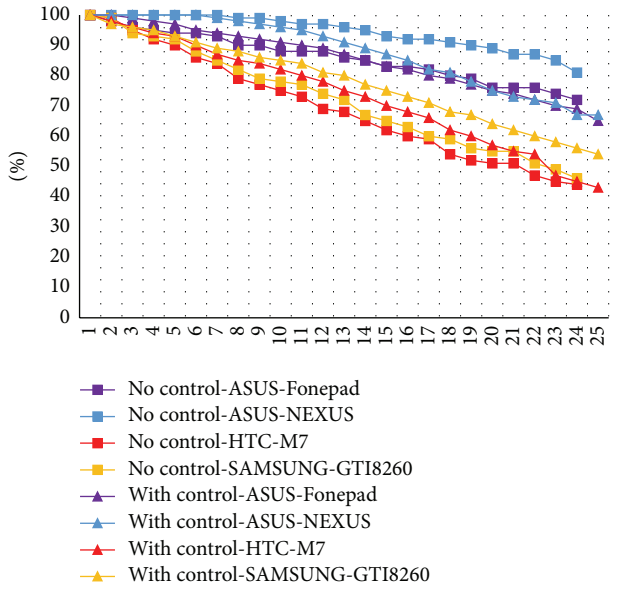


Figure 9: Energy-aware balancing result. x-axis: round; y-axis: remaining energy.

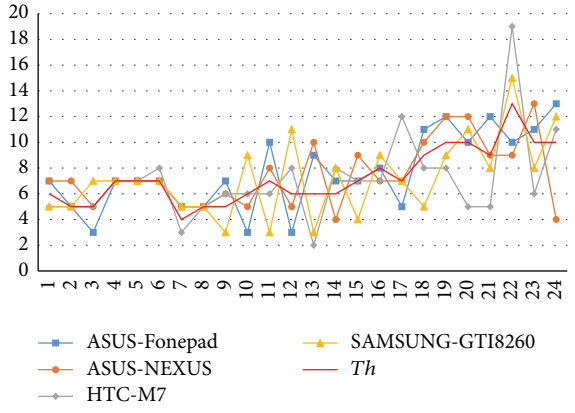


Figure 10: Data amount to energy proportions of individual devices (Xi) and the cloudlet federation (Th). x-axis: round; y-axis: data amount to energy proportions.

tiers such that mobile devices not only request services but also provide resources required by diverse services. The implemented HiBA architecture with feedback control framework proves improved performance when resources are adequately distributed to lower tier federations rather than being centralized in a remote cloud server. Through two case studies of cloudlet energy-aware balancing and bigger data center load balancing, we show that the HiBA is actually a development platform for mobile cloud computing and provides a feasible solution to UCN services. Future works include optimization algorithms embedding and big data analytics applications with crowd sensing are to be continued.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

Thanks are due to the Ministry of Science and Technology (MOST, which was the National Science Council) in Taiwan for sponsoring this research under continuous funded Projects NSC101-2221-E-327-020-, NSC101-2221-E-327-022-, NSC102-2218-E-327-002-, MOST103-2221-E-327-048-, MOST105-2221-E-327-027-, and MOST105-2221-E-006-141-.

References

[1] Technicolor, "User-Centric Networking," http://usercentricnetworking.eu/about-ucn/.

[2] G. Iosifidis, L. Gao, J. Huang, and L. Tassiulas, "Incentive mechanisms for user-provided networks," *IEEE Communications Magazine*, vol. 52, no. 9, pp. 20–27, 2014.

[3] B. A. A. Nunes, M. A. S. SAntos, B. T. De OliveirA, C. B. MArgi, K. ObrAczkA, and T. Turletti, "Software-defined-networking-enabled capacity sharing in user-centric networks," *IEEE Communications Magazine*, vol. 52, no. 9, pp. 28–36, 2014.

[4] G. Aloi, M. D. Felice, V. Loscrì, P. Pace, and G. Ruggeri, "Spontaneous smartphone networks as a user-centric solution for the future internet," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 26–33, 2014.

[5] F. Hao, M. Jiao, G. Min, and L. T. Yang, "A trajectory-based recruitment strategy of social sensors for participatory sensing," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 41–47, 2014.

[6] M.-L. Lee, H.-Y. Chung, and F.-M. Yu, "Modeling of hierarchical fuzzy systems," *Fuzzy Sets and Systems*, vol. 138, no. 2, pp. 343–361, 2003.

[7] T. C. Lin, M.-Y. Pai, C.-L. Chen, and C.-C. Chen, "Load-balanced cloud service interface for the HiBA mobile cloud environment," in *Proceedings of the IEEE International Conference on Consumer Electronics—Taiwan (ICCE-TW '15)*, pp. 360–361, Taipei, Taiwan, June 2015.

[8] C.-L. Chen, S.-C. Chen, C. Chang, and C. Lin, "Scalable and autonomous mobile device-centric cloud for secured D2D sharing," in *Information Security Applications*, vol. 8909 of *Lecture Notes in Computer Science*, pp. 177–189, 2015.

[9] W.-T. Su, W. Liu, C.-L. Chen, and T.-P. Chen, "Cloud access control in multi-layer cloud networks," in *Proceedings of the IEEE International Conference on Consumer Electronics—Taiwan (ICCE-TW '15)*, pp. 364–365, IEEE, Taipei, Taiwan, June 2015.

[10] H. Huang, H. Yang, and E. H. Lu, "A fuzzy-rough set based ontology for hybrid recommendation," in *Proceedings of the IEEE International Conference on Consumer Electronics—Taiwan (ICCE-TW '12)*, pp. 358–359, Taipei, Taiwan, June 2015.

[11] C.-L. Chen and C.-T. Chen, "Hierarchical Brokering with feedback state observation in Mobile Device-Centric Clouds," in *Proceedings of the 7th International Conference on Ubiquitous and Future Networks (ICUFN '15)*, pp. 410–415, July 2015.

[12] C. J. S. Decusatis, A. Carranza, and C. M. Decusatis, "Communication within clouds: open standards and proprietary protocols for data center networking," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 26–33, 2012.

[13] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[14] S. Mahadev, "Mobile computing: the next decade," in *Proceedings of the the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys'10)*, pp. 1–6, San Francisco, Calif, USA, June 2010.

[15] C.-L. Chen, J.-W. Lee, W.-T. Su, M.-F. Horng, and Y.-H. Kuo, "Noise-referred energy-proportional routing with packet length adaptation for clustered sensor networks," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 3, no. 4, pp. 224–235, 2008.