IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 14, 2020, accepted May 26, 2020, date of publication June 4, 2020, date of current version June 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2999938

CFRO: Cloudlet Federation for Resource Optimization

MUHAMMAD ZIAD NAYYER[1,2], IMRAN RAZA[2], (Member, IEEE), AND SYED ASAD HUSSAIN[2]

[1]Department of Computer Science, GIFT University, Gujranwala 52250, Pakistan
[2]Department of Computer Science, COMSATS University Islamabad, Lahore Campus, Lahore 54000, Pakistan

Corresponding author: Muhammad Ziad Nayyer (ziadnayyer@gmail.com)

ABSTRACT A Cloud computing paradigm augments the limited resources of mobile devices resulting in increased distance, limited Internet bandwidth, and seamless connectivity challenges between a remote cloud and mobile devices. Cloudlet computing based solutions are widely used to address these challenges by bringing the computational facility closer to the user. The ever growing number of mobile devices, Internet of Things (IoT) sensors and Information Communication Technology (ICT) infrastructure used for smart cities demand more resources. The existing cloudlet based solutions are unable to manage the ever-increasing demand for power, storage, and computational resources, and therefore forward the resource extensive tasks to a remote cloud, limiting cloudlet computing benefits. We present the Cloudlet Federation for Resource Optimization (CFRO), a federated cloudlet model for resource optimization to address these resource scarcity challenges. The proposed model exerts the features of scalability, resource collaboration, and robustness. The underlying scheme for resource optimization has been modeled as a Nested Multi Objective Resource Optimization Problem (NMOROP) and a novel algorithm has been proposed to solve it. The detailed analysis and comparative results show that the proposed model offers improved performance and more resource elasticity as compared to the conventional cloudlet model.

INDEX TERMS Cloudlet computing (CC), cloud federation (CF), fog computing (FC), Internet of Things (IoT), mobile cloud computing (MCC), mobile edge computing (MEC), smart cities (SC).

I. INTRODUCTION

Mobile devices are considered resource constrained due to limited computational, energy, and storage resources. On the other hand, resource intensive applications being developed regularly add more to the problem. Additional requirement of resources leads to the necessity of resource rich environment to offload compute intensive tasks. The cloud computing paradigm has addressed this necessity by offering extensive computational, energy and storage resources [1]–[3]. However, challenges such as distance, bandwidth, and seamless connectivity between remote cloud and mobile devices faced by conventional Mobile Cloud Computing (MCC) model limit its usage [4]. Mobile Edge Computing (MEC) [5], [6], Fog Computing (FC) [7], [8], and cloudlet computing [9] overcome these challenges by moving the computational facility at the edge of the core network and in the closer proximity of the user [10], [11]. This paper is focused on cloudlet based solutions as these offer more diversified features and generalized services with less dependency over the Internet [12]. Cloudlet is a mini cloud in the closer proximity of the user preferably in the same Local Area Network (LAN) at one hop distance with a stable Internet connection [13]. Mobile devices in the nearby vicinity can get an advantage from this mini cloud by offloading compute intensive tasks over Wireless Fidelity (WiFi) without Internet dependency.

The diversified features and generalized services make it more attractive for IoT devices and SC infrastructure [14]. However, with the increased workload, number of queries and devices, the cloudlet based solutions face resource scarcity challenges. The request is forwarded to a remote cloud to cope with the resource demand thus inducing the challenges of distance, limited bandwidth, and latency like the mobile cloud computing model. Hence, resource scarcity challenges at a cloudlet need to be addressed in a way that fewer requests are forwarded to the remote cloud [15]. The optimal solution must consider workload placement and load balancing to achieve resource optimization and scalability.

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

In this paper, we present the Cloudlet Federation for Resource Optimization (CFRO) model that offers scalability and resource optimization considering workload placement and load balancing. In the proposed model, the cloudlets form a federation to share the load and resources. The concept of federation that exists in cloud computing is entirely different from the cloudlet computing. Currently, most of the literature covers different topics in cloud federation [16]–[19]. The detailed architecture of the federation provided by NIST [20], [21] is also for cloud, where different cloud providers form a federation for resource sharing and a central broker that belongs to a third party acts as an intermediator. However, cloudlets have different challenges than cloud, and formation of a federation at the cloudlet level demands a different design and control mechanism. The proposed federated model is based on cloudlets that exist in closer proximity (Local Area Network or Metropolitan Area Network), and the broker exists on the cloud being part of the same federated model. The broker maintains the resource information of all the participant cloudlets. Using the aggregated information, the optimal placement, load balancing, and resource sharing decisions are taken. The proposed model considers resource optimization as a nested resource optimization problem, where an already placed Virtual Machine (VM) [22], [23] is also considered for re-optimization triggering a migration operation while enhancing the optimal solution space for new VM placement. Hence, both the placement and migration operations are considered as a part of the single bigger problem of resource optimization.

The proposed model has been evaluated using a live production environment consisting of a datacenter with around 20 physical servers, 65 Virtual Machines (VMs) and 4 Amazon EC2 instances. The datacenter has 3 Internet links from different ISPs. The results show that the proposed model can achieve better performance in terms of maximizing the resources and minimizing the total delivery time as compared to the conventional cloudlet model.

The major contributions of this paper are as follows:

- To the best of our knowledge, the concept of cloudlet federation has not been reported in the literature so far that addresses the resource scarcity challenges at a cloudlet with the objective of resource optimization while considering both the placement and migration operations as a part of the single problem.
- A time-aware allocation model called CFRO has been proposed to address the resource scarcity challenges at a cloudlet. CFRO offers scalability and load balancing features to provide resource optimized solutions.
- This paper then presents an implementation of a testbed namely ClPyZ that is available online [24]. It is built using an open source library called PsUtil. The underlying infrastructure is based on real time physical computers using Ubuntu Linux that can be easily modified and used for further research and development purposes.

The rest of the paper is organized as follows. Section II discusses the related work, followed by the system model and problem formulation in Section III. The proposed time-aware placement and migration algorithms are discussed in Section IV. Performance evaluation of the proposed CFRO model is presented in Section V. Section VI concludes the paper with open research challenges and future directions.

II. RELATED WORKS

This section provides an insight into the existing cloudlet based mobile augmentation approaches with the intent to evaluate possible solutions addressing resource scarcity challenges at the cloudlet level. The schemes presented in [25]–[29] are focused on the improvement of performance by reducing delay, CPU load and energy consumption. For delay improvement, the assumptions of static node and maximum hop count of two hops have been considered. However, if these two assumptions are not fulfilled, the proposed schemes perform poorly compared to cloud based scheme thus limiting their scope. For performance improvement, a queue based network scheme is introduced that addresses the following two challenges, 1) user to cloudlet assignment, and 2) K cloudlet placement, where K is an integer having the condition $K \geq 1$. For minimization of CPU load, a middleware platform has been proposed. It has the capability to deploy, remove, and change different software components on runtime. Thus, only required components are deployed reducing the load on CPU. However, the solutions offered by these approaches require some fundamental changes in the network design that are hard to realize due to cost and adaptability challenges.

The schemes presented in [30]–[32] are focused on the optimization of VM based cloudlet solutions. The factors of speedy VM instantiation, hand-off, and keeping the VM in the closer proximity of the user have been considered in these schemes. The closer proximity is achieved with an adaptive VM handoff technique that compares the existing VM state at the destination with the updated state at a source. The difference is evaluated and encoded with the delta approach [33]. For closer proximity, an integrated mobile network solution (MobiScud) based on Software Defined Networks (SDNs) and cloud has been proposed. The cloud is established in the closer proximity of a Radio Access Network (RAN) where customer's VMs are instantiated. Control messages between mobile devices and SDNs are monitored by MobiScud which are used to move the VM with the user to keep it in the closer proximity. However, the solutions offered by these schemes lack in providing support for some important aspects such as movement from RAN to WiFi, workload sharing and load balancing.

The techniques presented in [34], [35] are focused on Peer-to-Peer (P2P) communication based ad-hoc cloudlets. Factors of bandwidth consumption, offload latency, and resource constraints have been considered in these papers. Ad-hoc cloudlets enable a mobile device, having computational tasks, to access other devices in the closer proximity for resources. This cooperative scheme of workload sharing provides better results in terms of reduced bandwidth consumption,

offload latency, and resource constraint as compared to traditional cloud computing. However, these techniques are unable to cover diversified use cases such as sharing between more than two mobile devices thus limiting their scope.

In [36], [37], the utility maximization of participant devices is focused to form a cloudlet. A point system or virtual currency system is used to quantify incentives based on credit and reputation. Points are given to a device when its resources are utilized by other devices. A device may lose points when using community resources. However, these approaches have not been practically tested, hence their adaptability and efficiency cannot be determined.

The solutions presented in [38], [39] have the objective of providing seamless connectivity. Features of Wireless Mesh Networks (WMN) and context awareness have been used to achieve the objective of seamless connectivity. WMN offers a redundant path to reach a certain cloudlet and context awareness provides adaptability to a changing environment thus delivering seamless services to the users. However, the size of the mesh is very limited and context awareness works on closer proximity principle, hence these solutions are not suitable for larger networks.

The approaches in [40], [41] aims to minimize the cost in terms of energy and resource consumption. The objective of cost minimization is achieved through a cloning based scheme where each mobile device has a software clone in the cloud to offload compute intensive data. The other approaches include task partitioning and offloading the only part that is compute intensive or selection of most energy efficient access point, route, and cloudlet [42].

An overview of the capabilities offered by these solutions leads to the following conclusions:

- The resource scarcity challenge at cloudlet due to the increased number of devices, workloads, IoT sensors, and Smart City infrastructures has not been addressed.
- Resource brokerage required to work in a transparent manner considering fair resource sharing for cloudlet based solutions is not available.
- There is no such scheme that facilitates the entire cycle of resource management among participants for better assessment, allocation, and optimization.
- The resource optimization problem has not been viewed as a nested resource problem.

These findings are indicators of the necessity to devise a novel framework to address the resource scarcity challenges at the cloudlet for performance improvement.

III. SYSTEM OVERVIEW AND PROBLEM FORMULATION

A. SYSTEM MODEL

In the proposed federated cloudlet model shown in Figure 1, cloudlets form a federation with the intent to scale resources while minimizing offloading, request processing, execution, and migration time. Hence, the proposed model has the strengths of both MCC and cloudlet based models i.e. closer proximity and scalability. The users get the advantage of federated resources without the need to worry about the delay,

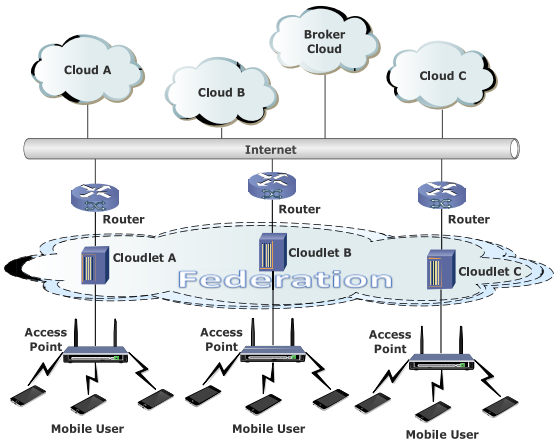

FIGURE 1. Federated cloudlet model.

resource constraints and dependency over the Internet as they get services from the immediate cloudlet present in the same WiFi network, preferably one hop away.

A central broker manages all the cloudlet federation operations of cloudlet membership, resource information management, and optimal selection. The request is forwarded to the broker by an immediate cloudlet and optimal decision is sent back consisting of a target cloudlet present in the federation. As only the request is forwarded and not the data, the effect on offloading time is negligible. On the other hand, the target cloudlet is present in the same Metropolitan Area Network (MAN) reducing offloading and migration time.

B. SYSTEM ARCHITECTURE

The high-level architecture of the proposed federated cloudlet model uses a modular approach for the performance optimization of mobile cloud computing. There are four participants in the proposed federated cloudlet model as shown in Figure 2.

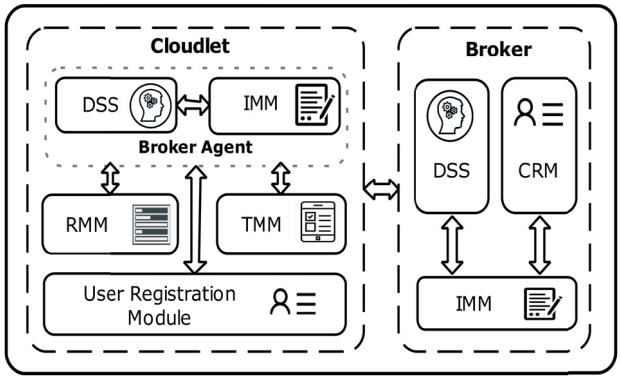

FIGURE 2. CFRO architecture.

Two of these are external i.e. i) Mobile Device ii) Remote Cloud and the other two are internal i.e. i) Cloudlet ii) Broker. The broker contains three modules i.e. i) Cloudlet Registration Module (CRM): Responsible for the registration of cloudlet ii) Decision Support System (DSS): Responsible for the selection of optimal cloudlet and VM for migration

and iii) Information Management System (IMS): Responsible for recording resources, performance parameter values and decision information.

The Cloudlet contains three modules i.e. i) Resource Management Module (RMM): Responsible for resource allocation and deallocation ii) Task Management Module (TMM): Responsible for task initiation, execution and completion, iii) User Registration Module (URM): Responsible for the registration of the user, and an auxiliary broker-agent acting on behalf of broker for the synchronization of information with broker stored by IMS. Each module is controlled by a module manager responsible for all activities within that module. Every module consists of different components that are invoked by incoming or outgoing requests. Inter-module communication is handled by module managers, whereas Intra module communication is handled directly by the services.

C. PROBLEM FORMULATION

Let's assume a user with T tasks bundled as a VM on a mobile device and needs to offload the VM on the cloudlet. The set of users is represented as $U = \{u_1, u_2, u_3, \ldots u_m\}$, set of VMs to be placed as $V = \{v_1, v_2, v_3, \ldots v_g\}$, set of cloudlets as $C = \{c_1, c_2, c_3, \ldots c_n\}$, set of required resources for VMs to be placed as $\mathcal{R} = \{r_1, r_2, r_3, \ldots r_l\}$, and class of required resources as $\mathcal{R}'$. The set of already placed VMs are represented as $V' = \{v'_1, v'_2, v'_3, \ldots v'_p\}$, set of required resources for already placed VMs as $\mathfrak{R} = \{r'_1, r'_2, r'_3, \ldots r'_q\}$, and class of required resources as $\mathfrak{R}'$. The set of available resources on a cloudlet are represented as $A = \{a_1, a_2, a_3, \ldots a_s\}$ and class of available resources as A'. The set of total resources on a cloudlet are represented as $\mathbb{R} = \{t_1, t_2, t_3, \ldots t_y\}$ and class of total resources as $\mathbb{R}'$. The set of occupied resources at a cloudlet are represented as $O = \{o_1, o_2, o_3, \ldots o_z\}$, set of tasks as $X = \{x_1, x_2, x_3 \ldots x_h\}$, and class of tasks as X'. A union set that contains all the VMs is defined as $V'' = V \cup V'$.

Definition 1: A relationship $U \rightarrow V$ is said to be one to many if an element of U is related to two or more elements of V. In MCC, a user to VM and further VM to task relationship of one to many can be defined as $V \rightarrow X$

A user may have multiple VMs and each VM v_f can have multiple tasks x_i defined by the following condition

$$v_f = \left\{ \sum_{i=1}^{k} x_i | i = 1 \; to \; k \right\} \tag{1}$$

such that

$$\forall \; users \; u_i, \; if \; u_i \; \exists \; some \; VM \; v_{ij}$$
$$where \; u_i \neq u_j$$
$$for \; i, j, k = 1, 2, 3, \ldots, n$$

Please note that the converse of the above conditions does not hold. An undirected graph $G(V, E)$ as presented in Figure 3 represents a connection between the mobile user and cloudlets. An edge E represents a connection between a user u_i and cloudlet c_i such that $(u_i, c_i) \in E$. There are three kinds of vertices in V. A vertex representing a mobile user u_i, a vertex representing cloudlet with insufficient resources c_i and a vertex representing sufficient resources c_j, where both $c_i, c_j \in C$. A tuple $(\alpha_i, \beta_i, \gamma_i)$ represent a request where α_i represents a user $u_i \in U$, β_i represents the set of required resources $\mathcal{R}_i \in \mathcal{R}'$, and γ_i represents a VM $v_i \in V$. Rest of the notations used throughout the paper are presented in Table 1.

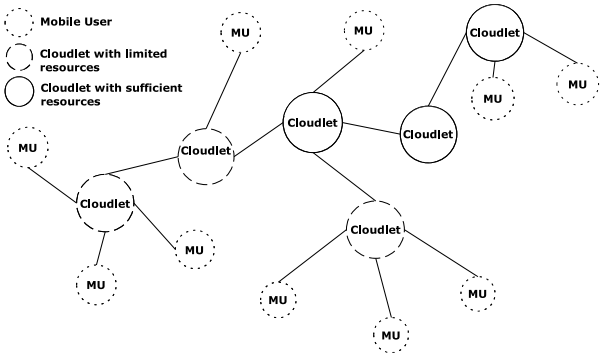

FIGURE 3. User to cloudlet graph.

TABLE 1. Table of notations.

Notation	Definition
φ	conventional cloudlet computing model
$\aleph$	CFRO
S_f	VM size
$\emptyset$	bit rate
τ	throughput
L	latency
T_{dt}	total delivery time
t_{tr}	transmission time
t_{pd}	propagation delay
t_{rp}	request processing time
t_e	execution time
$\mathcal{R}$	required resources for new VM placement
$\mathfrak{R}$	required resources for already placed VM
$\mathbb{R}$	Total resources of a cloudlet
$\mathcal{R}'$	class of required resources for already placed VM
A	set of available resources
V	set of VMS to be placed
V'	set of already placed VMs
V''	total VMs on a cloudlet
S_D	request decision status
Θ	optimal decision
c	cloudlet
c_e	eligible cloudlet
c_o	optimal cloudlet
h	cloudlet hop-count
v_O	optimal VM for migration
v_E	eligible VM for migration

It is assumed that the capacity of a cloudlet to host several VMs can be calculated by comparing available resources to the required resources. However, the status of available resources changes every time a VM is launched or removed. A resource set occupied by a VM represents a slot and multiple slots can be available on a single host. Let's consider a

scenario where there are k VMs with $\mathcal{R}'$ required resources and n' available slots on cloudlets. Since, there are multiple cloudlets in the federation and to offload a VM, an optimal cloudlet c_i is selected such that it meets all of its resource demands while keeping the total delivery time T_{dt} minimum. Hence, the first objective function to minimize T_{dt} for all VM placements with constraints can be formulated as follows

$$\begin{gathered} \forall\ VMs\ v_k \in \mathrm{V} \\ \min T_{dt}\left(\sum_{i=1}^{n} v_i\right) \\ for\ i,k = 1,2,3,\ldots,n \end{gathered} \tag{2}$$

Constraint 1: A VM can only be placed on a cloudlet, if the available resources on a cloudlet meet the required resources.

$$\begin{gathered} \forall\ resources\ a_i \in A\ and\ r_i \in \mathcal{R} \\ Place\ VM\ v_z\ if\ \sum_{i=1}^{n} a_i \geq r_i \\ then\ \exists\ some\ cloudlet\ c_k\ to\ host\ the\ VM\ v_z \\ for\ i,k,z = 1,2,3,\ldots,n \end{gathered} \tag{3}$$

Constraint 2: Total required resource capacity of all the VMs must not exceed the total resource capacity of the cloudlet.

$$\begin{gathered} \forall\ resources\ r'_i \in \Re\ and\ t_i \in \mathbb{R} \\ \sum_{i=1}^{n} r'_i \leq t_i \\ for\ i = 1,2,3,\ldots,n \end{gathered} \tag{4}$$

However, it has already been shown that both placement and migration operations are interdependent and the first can trigger the second. Hence, the ultimate decision is forecasted by re-optimizing the whole objective function. Let's take an example scenario where there are k VMs v_i to be placed with R required resources and k' VMs v'_j already placed with R required resources and n' available slots on cloudlets. There are multiple cloudlets in the federation and to offload a VM, an optimal cloudlet c_i is selected such that the total delivery time T_{dt} is minimum. The Optimal Cloudlet Selection (OCS) process considers to migrate an already placed VM on c_i closer to its source cloudlet thus creating room for the new request for VM placement. If such a possibility exists such that it re-optimizes the objective function for both already placed VM and new requests for VM placement, both migration and placement operations are executed. Hence, the nested objective function to minimize T_{dt} for all VM placements and migrations with constraints can be formulated as follows

$$\begin{gathered} \forall\ VMs\ v_k, v'_j \in V'' \\ \min T_{dt}\left(\sum_{i,j} v_i + v'_j\right) \\ for\ i,j,k = 1,2,3,\ldots,n \end{gathered} \tag{5}$$

Constraint 3: Re-optimization of VM placement decision for a new request is only possible if an already placed VM can be migrated in such a way that it re-optimizes both decisions for already placed VM and new VM.

$$\begin{gathered} \forall\ resources\ a_i \in A, r_i \in \mathcal{R}\ and\ r'_i \in \Re \\ Migrate\ VM\ v'_j\ if\ \sum_{i=1}^{n} a_i \geq r'_i \\ then\ \exists\ some\ cloudlet\ c_k\ to\ host\ the\ VM\ v'_j \wedge \\ Place\ VM\ v_z\ if\ \sum_{i=1}^{n} a_i \geq r_i \\ then\ \exists\ some\ cloudlet\ c_i to\ host\ the\ VM\ v_z \\ for\ i,j,k,z = 1,2,3,\ldots,n \end{gathered} \tag{6}$$

Constraint 4: A VM with hop-count $h' \geq 2$ satisfying the resource demand of a new VM is only considered for migration, since VMs with $h' < 2$ are assumed to be already placed nearest to the user with minimum T_{dt} and cannot be further optimized. The hop-count for new VM and an already placed VM is represented by h and h' respectively.

$$\begin{gathered} \forall\ resources\ a_i \in A\ and\ r'_i \in \Re \\ if\ \sum_{i=1}^{n} a_i \geq r'_i \wedge h'_j \geq 2 \\ then\ \exists\ some\ VM\ v'_j\ for\ migration \end{gathered} \tag{7}$$

IV. TIME-AWARE RESOURCE OPTIMIZATION

The proposed approach offers a resource-aware collaborative scenario in which there are multiple options for offloading sites. The optimal site is the one that provides minimum total delivery time T_{dt} considering the resource demand. These options may include local cloudlet, neighboring cloudlets, and remote cloud. The collaborative approach requires to share resource information of each cloudlet to filter eligible cloudlets for a particular request, and then selects the optimal cloudlet based on available resources and total delivery time. On the other hand, this approach also keeps track of the cloudle's resource level for load balancing and optimal VM selection for migration. A simple flow of information is shown in Figure 4.

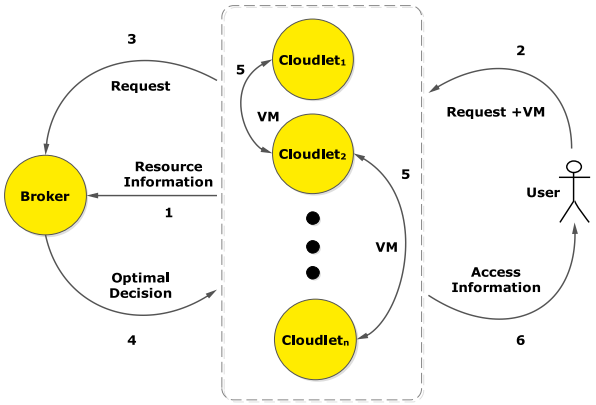

FIGURE 4. CFRO data flow.

A user α_1 initiate a request requiring β_1 resources and offload VM γ_1 to the cloudlet. The request is forwarded to the broker for optimal cloudlet decision. Broker keeps track of all member cloudlets and pushes the decision back to a source cloudlet. The VM is on the optimal cloudlet and is executed.

An ideal condition is where the source cloudlet is the optimal one, thus not requiring any further calculation.

A. RESOURCE CALCULATION

Le's assume that a request is received by $\boldsymbol{c_1}$ and forwarded to the broker. The total, available, occupied, and required resources at cloudlet $\boldsymbol{c_1}$ can be defined as follow

$$\mathbb{R}(c_1) = \begin{bmatrix} r''_1 \\ r''_2 \\ r''_3 \\ r''_4 \end{bmatrix}, \quad A(c_1) = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}, \; O(c_1) = \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix},$$

$$\mathcal{R} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}$$

These resource matrices are pushed to broker from time to time and the broker maintains a record of all these resources for every member cloudlet. For example a tuple (t_1, a_1, o_1) represents the quantity of total, available, and occupied CPU resource, (t_2, a_2, o_2) represents the quantity of total, available, and occupied memory resources, (t_3, a_3, o_3) represents the total, available and occupied hard disk drive resource, and (t_4, a_4, o_4) represents the total, available and occupied bandwidth resource. The total resources of a cloudlet can be calculated as follow

$$\mathbb{R}(c_i) = A(c_i) + O(c_i) \tag{8}$$

and

$$A(c_i) = \mathbb{R}(c_i) - O(c_i) \tag{9}$$

There can be two cases of a request for VM placement

Case 1: If c_1 has enough resources to execute the request such that

$$\begin{gathered} \forall \; resources \; a_i \in A, r_i \in \mathcal{R} \\ if \; a_i > r_i \; then \; \exists \; some \; cloudlet \; c_k \\ for \; i, k = 1, 2, 3, \ldots, n \end{gathered} \tag{10}$$

Case 2: If c_1 does not have enough resources to execute the request such that

$$\begin{gathered} \forall \; resources \; a_i \in A, r_i \in \mathcal{R} \\ if \; a_i < r_i \; then \; \nexists \; some \; cloudlet \; c_k \\ for \; i, k = 1, 2, 3, \ldots, n \end{gathered} \tag{11}$$

The probability that c_1 has adequate resources to execute the job is p and inverse of this condition can be formulated as 1-p. The case of not having enough resources is significant as it leads to the choice of forwarding the request to a remote cloud. So, the probability that c_1 does not have enough resources to execute the job and can forward the request to the remote cloud can be defined in terms of conditional probability problem.

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} \tag{12}$$

where P is the probability function, A is the probability that c_1 does not have adequate resources to execute the job and B represents the probability p_c that c_1 can forward the request to the remote cloud. The probability that c_1 is not connected with the remote cloud is ignored as there is no other option for the request to wait for available resources at the cloudlet c_1. Hence the conditional probability can be formulated as follows

$$P(B \mid A) = \frac{P(1 - p \cap p_c)}{P(1 - p)} \tag{13}$$

B. TIME CALCULATION

The total delivery time is dependent upon latency L, offloading time t_o, migration time t_m, and request processing delay t_{prd}. Latency is measured as Round Trip Time (RTT) and is directly dependent upon hop-count. Offloading time t_o refers to the time required for offloading the VM from mobile device to the cloudlet and is the combination of transmission time ot_{tr} and propagation delay ot_{pd}. Transmission time is dependent upon file size § and bit rate Ø. There is an inverse relationship between file size and bit rate. However, the link condition, capacity, and receiving devic's capability to receive a transmitted signal often reduces the transfer rate and thus maximum achievable rate is defined in terms of throughput τ and can be calculated by replacing bit rate Ø with throughput τ. Propagation delay t_{pd} is the amount of time required by the signal to travel distance d with velocity v from source to destination. There is an inverse relationship between distance d and velocity v. Migration time t_m is the amount of time required by the VM to migrate from source cloudlet to optimal cloudlet and can be calculated in a similar way as offloading time in Equation (17).

Request processing delay t_{prd} is the amount of time required by the cloudlet to process the request including the time spent in the wait queue t_w and execution time t_e. The wait time spent in the queue is dependent upon the task arrival rate that can be obtained by the statistical analysis of the host server and follows a poison distribution, identified by λ. A single cloudlet can serve a single request at a time, therefore, given the number of cloudlets c_i, the average task arrival rate Λ is given by the sum of arrival rate at all individual cloudlets [43]. Each cloudlet have k processers and every processor executes the task following an exponential distribution with an average execution time $t_e = 1/\mu$. These assumptions leads to a M/M/k queue model [43]. The occupation rate O_r per processor is $\lambda/(k \times \mu)$. Now the probability that a task has to wait in the queue is as follows

$$p = \frac{(k \times O_r)^k}{k!} \times \left((1 - O_r) \times \sum_{m=0}^{k-1} \frac{(k \times O_r)^m}{m!} + \frac{(k \times O_r)^k}{k!} \right)^{-1} \tag{14}$$

As per probability p, the average waiting time for a task in a queue is as follows

$$t_w = p \times (1 - O_r)^{-1} \times (k \times \mu)^{-1} \tag{15}$$

The total average processing delay per task is the sum of wait time and execution time, and is formulated as follow

$$t_{prd} = p \times (1 - O_r)^{-1} \times (k \times \mu)^{-1} + \frac{1}{\mu} \tag{16}$$

Hence, the total delivery time can be formulated as follows

$$\begin{aligned} T_{dt} &= ot_{tr} + ot_{pd} + mt_{tr} + mt_{pd} + t_w + t_e \\ T_{dt} &= o\left(t_{tr} + t_{pd}\right) + m\left(t_{tr} + t_{pd}\right) + t_w + t_e \\ T_{dt} &= o\left(\frac{\S}{\emptyset} + \frac{d}{s}\right) + m\left(\frac{\S}{\emptyset} + \frac{d}{s}\right) + p \times (1 - O_r)^{-1} \\ &\quad \times (k \times \mu)^{-1} + \frac{1}{\mu} \\ T_{dt} &= o\left(\frac{\psi}{\tau} + \frac{d}{s}\right) + m\left(\frac{\psi}{\tau} + \frac{d}{s}\right) + p \times (1 - O_r)^{-1} \\ &\quad \times (k \times \mu)^{-1} + \frac{1}{\mu} \end{aligned} \tag{17}$$

C. OPTIMAL CLOUDLET SELECTION

The Optimal Cloudlet Selection process is divided into two phases. In the first phase, eligible cloudlets having adequate resources for the requested job are filtered using the following conditio

$$f(x) = \begin{cases} 1, & if\ \mathcal{R}[r_i] \leq A\left[a_{ij}\right] \\ \forall & \begin{cases} i = 1, & j = 1, 2, 3 \ldots n \\ i = 2, & j = 1, 2, 3 \ldots n \\ i = 3, & j = 1, 2, 3 \ldots n \\ & \cdot \\ & \cdot \\ & \cdot \\ i = n, & j = 1, 2, 3 \ldots n \end{cases} \\ 0, & Otherwise \end{cases} \tag{18}$$

The value of ''1'' represents eligible cloudlets having adequate resources and ''0'' represents non-eligible cloudlets without adequate resources to execute the task. In the second phase, total delivery time T_{dt} for each cloudlet is calculated using Algorithm 1.

The optimal selection algorithm returns optimal cloudlet c_O with minimum hop-count h and T_{dt}. The value of hop-count is observed to see if a VM is already placed at Hop-count 1, this means that it is already placed nearest to source and its placement cannot be further optimized. The time complexity of algorithm 1 is $O(n^3)$.

D. NESTED OPTIMIZATION

The next step in NMOROP is nested optimization presented as Algorithm 2 with a time complexity of $O(n^5)$.

Algorithm 1 Optimal Cloudlet Selection (OCS)

Input: List of requests at a broker, List of required resources for each request, list of cloudlets, list of available resources at cloudlet, list of cloudlet hop-count for each request
Output: Optimal cloudlet c_O, Hop-count h and T_{dt}

```
Begin:
  Let optimal cloudlet c_o be NULL
  for each request in request list at broker do
      check status S_D
      if S_D = "Decision Pending"
          for each cloudlet c_i in cloudlet list do
              for each resource a_i and r_i in available and
              required resource list do
                  if a_i ≥ r_i
                  push cloudlet c_i in eligible cloudlet list
                  end if
              end for
          end for
          for each cloudlet in eligible cloudlet list do
          calculate h and T_dt
          end for
      end if
      c_O = c_i with minimum T_dt
  end for
  return c_O, h(c_O), T_dt(c_O)
end:
```

The nested optimization process can be divided into four phases. In the first phase, it receives a request and optimal decision parameters from OCS, and tries to identify a VM in the closest proximity whose resource requirement matches with the one in this request. The identified VM must not already be placed at the source cloudlet with a hop-count of 1. If that is the case, then it is already placed at the nearest location and its total delivery time cannot be further optimized.

In the second phase, it sends the identified VM's list to OCS for a second iteration and get results. It further compares the results returned by OCS against each VM in the list sent to OCS to see if a VM can be migrated to its source cloudlet or near to source thus re-optimizing total delivery time.

In the third phase, by following a greedy approach if it finds such VM in the closest proximity that can be migrated, it removes the resources occupied by this VM and add them to the available pool. The operation of removing resources and adding them to the available pool is only at the information level used for the forecasting process.

In the fourth phase, it resends the new request for the second iteration to OCS and check if removing the resources occupied by already placed VM re-optimizes the decision and can the same cloudlet be selected as optimal cloudlet for a new request. The proposed time-aware approach based on

Algorithm 2 Nested Optimization (NO)

Input: List of requests at the broker, List of required resources for each request, list of cloudlets, list of available resources at cloudlets, list of cloudlet hop-count for each request, list of running VMs, list of occupied resource by each VM, list of optimal cloudlet hop-count, list of optimal T_{dt} for each request

Output: request, decision

Begin:
Let decision Θ be *NULL*
for each request in request list at broker **do**
  check status S_D
  if $S_D =$ "*Decision Pending*"
    for each cloudlet c_i in cloudlet list **do**
      if $h \geq 1$ *and* $< h(c_o)$
      push cloudlet c_i in eligible cloudlet list
      end if
    end for
  end if
  for each cloudlet c_i in eligible cloudlet list
    for each VM in running VM list **do**
      for each resource r_i and r'_i in the required resource list for new VM and required resource list for already placed VM **do**
        if $r_i \leq r'_i$ *and* $h(v'_i) \geq 1$
        push VM in eligible VM list
        end if
      end for
    end for
  for each VM in eligible VM list **do**
    send VM v'_i to algorithm 1 for the second iteration
    get results
    if $(h(v'_i) <$ *new* $h(v'_i) \wedge T_{dt}(v'_i) <$ *new* $T_{dt}(v'_i))$
      remove resources and send new VM v_i for the second iteration to algorithm 1
      get results
      if $(h(v_i) <$ *new* $h(v_i) \wedge T_{dt}(v_i) <$ *new* $T_{dt}(v_i))$
        $\Theta =$ migrate VM (v'_i) on $c_o \wedge$ place VM (v_i) on c_i
        $S_D =$ "*Decision Complete*"
      end if
    else
    $\Theta =$ place VM (v_i) on c_o
    end if
  end for
end for
return *request*, Θ
end:

cloudlet federation ensures optimal results i.e. min delivery time as compared to the conventional approach.

V. PERFORMANCE EVALUATION

In this section, performance analysis of the resource-aware allocation heuristic presented in Section IV has been discussed.

A. NESTED OPTIMIZATION

The testbed used to conduct the experimentations is ClPyZ as shown in Figure 5.

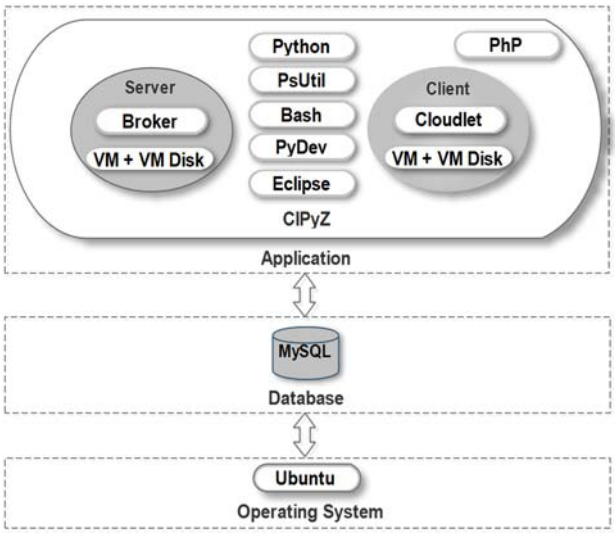

FIGURE 5. ClPyZ architecture.

It is an open-source virtualization platform developed as a part of this research. The reason for the development of this novel platform is the unavailability of a suitable environment to execute the proposed federated cloudlet model. The major requirements to execute a federated cloudlet environment include management of multiple clouds, client side resource monitoring and VM migration between cloudlets. The server-end of ClPyZ is developed in Python 2.7 language using the Eclipse PyDev extension and PsUtil library. The client-end has a web interface built-in PHP and the back-end database is developed using MySQL. It introduces a centralized brokerage system to manage multiple cloudlets belonging to different clouds.

ClPyZ addresses the heterogeneity problems of cloudlets by offering platform independent, flexible, and generalized features that can be customized or redefined as per need. It supports diversified virtualization file and disk formats and more can be added. ClPyZ has been extensively tested and is ready for production. It can be installed on any Linux distribution supporting standard shell programming for OS level operations.

The main features of ClPyZ include:

- Custom Orchestration: Any standard VM template can be used supporting Open Virtualization Format (OVF).
- Resource Tracking: The available resources for the federation environment are automatically tracked for resource provisioning.

- Monitoring: The resource monitoring and alert system is available for cloudlets to indicate under-provisioning or over-provisioning of resources.
- GUI for clients: Provides easy manageability for user requests and status monitoring.
- Multi-cloud management: Cloudlets belonging to different clouds can be added to the federation.
- Centralized Control and Visibility: The server side dashboard provides explicit monitoring, logging, and event based triggers for easy administration.
- Scalability and Recourse Sharing: The cloudlet federation resources can be scaled up by having more members and hence more possibility of resource sharing.
- VM Migration: A VM can be migrated from one cloudlet to another to optimize the resource utilization.

B. PERFORMANCE METRICS

To compare the efficiency of the proposed algorithms several performance metrics have been selected. The first metric is the latency L between source and destination. There are two cases, one is cloudlet-to-cloudlet and the second is cloudlet-to-cloud, where the cloud is a distant remote cloud. The second metric is throughput τ. The third metric is hop-count h between source and destination. The fourth metric is migration time t_m from the source to destination. The fifth metric is number of requests N_r before the worst case, where the worst case is when a cloudlet has to offload the task to the distant remote cloud due to the unavailability of the resources at a local or neighbor cloudlet.

C. EXPERIMENTAL SETUP

Since the proposed system is ready to work in a production environment, this gives an edge to use it for real time requests and take practical values of different observable parameters. A test-bed for both the conventional cloudlet model and the proposed federated cloudlet model has been set up. The conventional cloudlet model as shown in Figure 6 comprises a remote cloud and a cloudlet node in the closer proximity of the user, at one-hop distance.

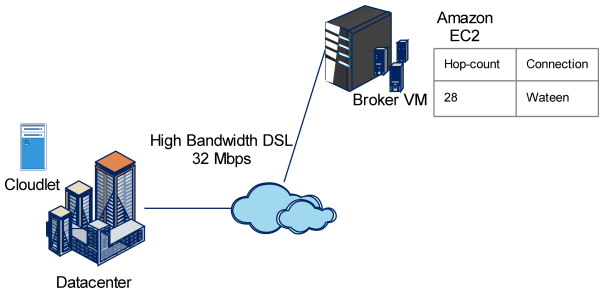

FIGURE 6. Conventional cloudlet setup.

Remote cloud instance has been created on the Amazon EC2 cluster and the cloudlet node is created in a virtualized environment. The virtualized environment has been deployed in a datacenter on HP DL360 G6 server with quad-core 2.8GHz Xeon processor, 64GB of RAM, 250GB of SAS drive, quad 1Gbps NICs having 32Mbps of bandwidth available for Internet interface. The virtualized environment has been developed using VMware ESX 6.0 server. Each cloudlet node is configured with a single CPU, 8GB of RAM, 30GB of storage, and Ubuntu 14.04 LTS operating system.

The proposed model has been developed using 7 nodes as shown in Figure 7. One broker node and 6 cloudlet nodes. Cloudlet nodes have been deployed at 3 different locations within the same city, two at each location. Cloudlet nodes are configured with 2 different ISPs. One node at each location belongs to an ISP, while the second node at each location belongs to other ISP. The use of two different ISPs enable us to observe the parameters for both MAN and WAN environments. The same virtualized environment and parameters have been used for both experimental setups. Moreover, the parameters presented as performance metrics are mandatory and do not hold any preference, sequence or weights as mentioned in equation (17).

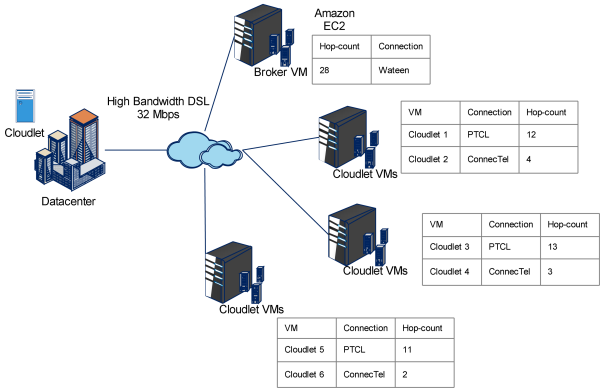

FIGURE 7. Proposed federated cloudlet setup.

A user gets registered at a cloudlet and forwards offloading requests. The user offloads the OVA file to the connected cloudlet. Tiny Core Linux (TCL) configured with 1 CPU, 48 MB of RAM, 45MB of storage having a size of 15 MB has been used for experimentations. The sample metric values for both experimental setups have been presented in Table 2.

TABLE 2. Sample metric values for study cases.

Study Case	1	2	3	4	5
File Size	10MB	15MB	20MB	25MB	30MB
Number of Request	15	30	45	60	75
Users	1	5	10	20	25
Bandwidth	32Mbps up/down				

In the conventional cloudlet model, if there are ample resources, the request is executed on a connected cloudlet. Otherwise, the request is forwarded to the remote cloud and this decision is taken by the cloudlet. In the proposed model, the request is forwarded to the broker by a cloudlet for optimal decisions using a resource-aware algorithm. The request is either executed on a connected cloudlet or any other cloudlet in the cloudlet federation. The place of decision does not

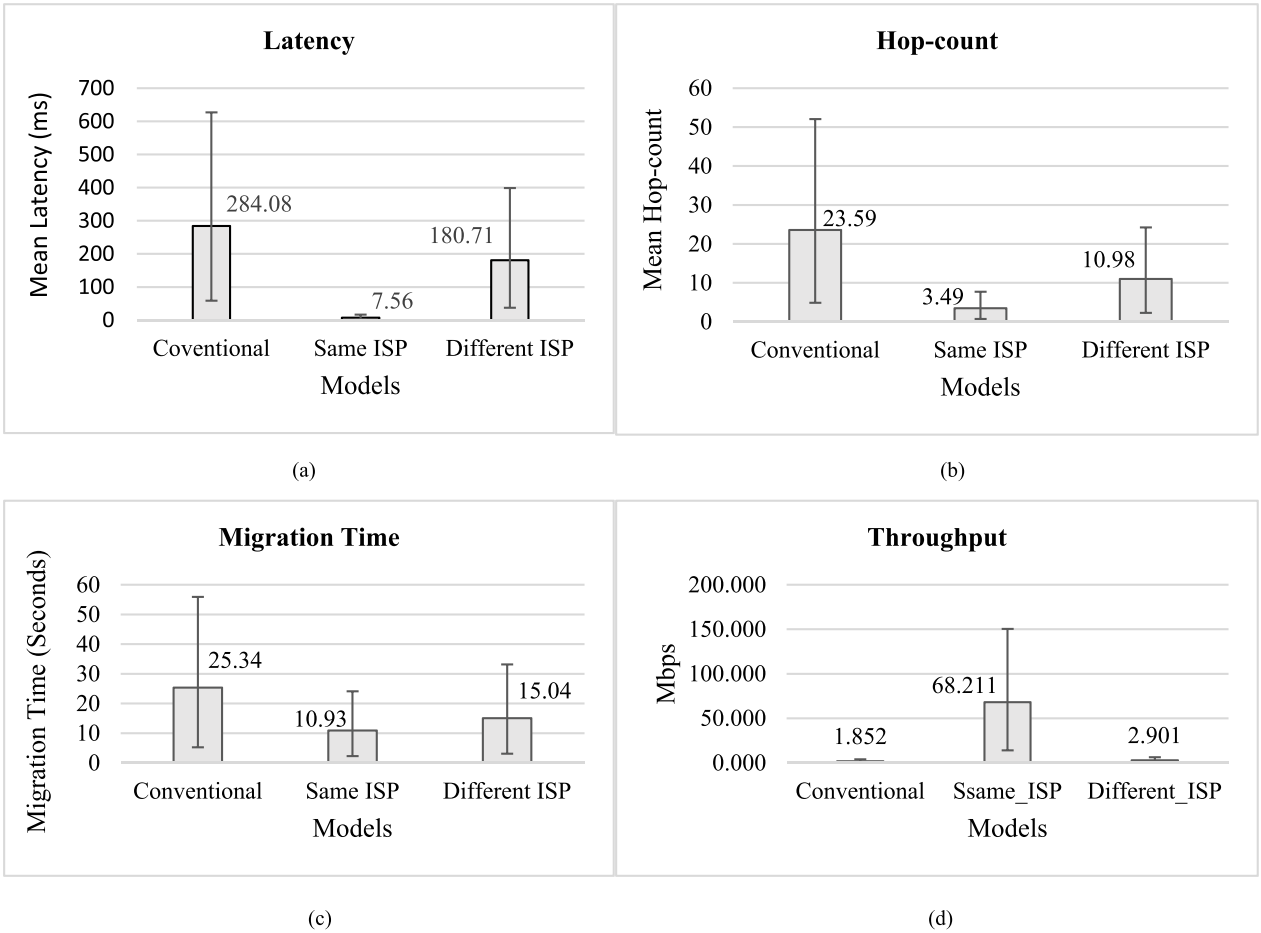

FIGURE 8. Analysis of latency, hop-count, migration time and throughput for conventional and CFRO models.

matter as the request latency is negligible. However, the request execution place is important due to VM size, required resource, and migration latency. The best case is the same for both setups as the request is executed on the connected cloudlet. However, the average and worst-case for both setups are different. In the conventional cloudlet model, the request is forwarded to the remote cloud in the worst case, whereas in the proposed model the worst case is never executed unless the whole cloudlet federation is out of resources which is hard to realize.

The main objective is to overload the system beyond available resources on a single cloudlet to observe the behavior of both conventional and federated cloudlet models. The overloading process is independent of the type and nature of the workload since only the required resources are considered. The experiments have been conducted in two phases. In the first phase using a conventional cloudlet model, some requests are initiated in the form of VMs until a cloudlet starts forwarding requests to the remote cloud due to the unavailability of resources. In the second phase, the same number of requests are initiated on the proposed model for comparative results. The process of initiating requests continues even after reaching the worst case to observe the behavior of both models.

D. TESTBED RESULTS

All tests were repeated at least three times for both conventional and proposed models. The values of performance parameters are summarized in Table 3 using average, standard deviation, and confidence (Cd) for a significance value of 0.05.

A VM using the Amazon EC2 instance has been instantiated that serves as a remote cloud for the conventional model. Two network use cases are used for the proposed model; 1) using the same ISP for source and destination cloudlets, 2) using different ISPs for source and destination cloudlets. The detailed results and findings in terms of different parameters and their relationship for the conventional and proposed model are as follows

1) Latency and Hop-count: These parameters have been evaluated for conventional and the proposed model under the worst case, where the connected cloudlet is out of resources. The results shown in Figure 8 indicate an elevated level of latency for the conventional model due to increased distance and hop-count as compared to the proposed model. Fixed bandwidth and VM size has been considered to observe the migration time. In the case of the same ISP, the reduction in terms of overall latency is 97% and in terms of hop-count it is by 91%.

TABLE 3. Overview of parameters values for both models.

Parameters		Avg	Std.Dev	Cd		Avg	Std.Dev	Cd		Avg	Std.Dev	Cd
Latency	Conventional	284.08	4.9066	0.9736	Same ISP	7.56	0.4989	0.099	Different ISP	180.71	1.1128	0.2208
Hop-count		23.59	3.4468	0.6839		3.49	0.5024	0.0997		10.98	0.5024	0.1715
Migration Time		25.34	2.0162	0.4001		10.93	0.8439	0.1675		15.04	0.0175	0.1593
Throughput		4.546	0.1306	0.0259		10.527	0.4224	0.0838		7.629	0.1998	0.0396
NRBW		27	0.7914	0.157		36.17	1.4003	0.2778		36.17	1.4003	0.2778

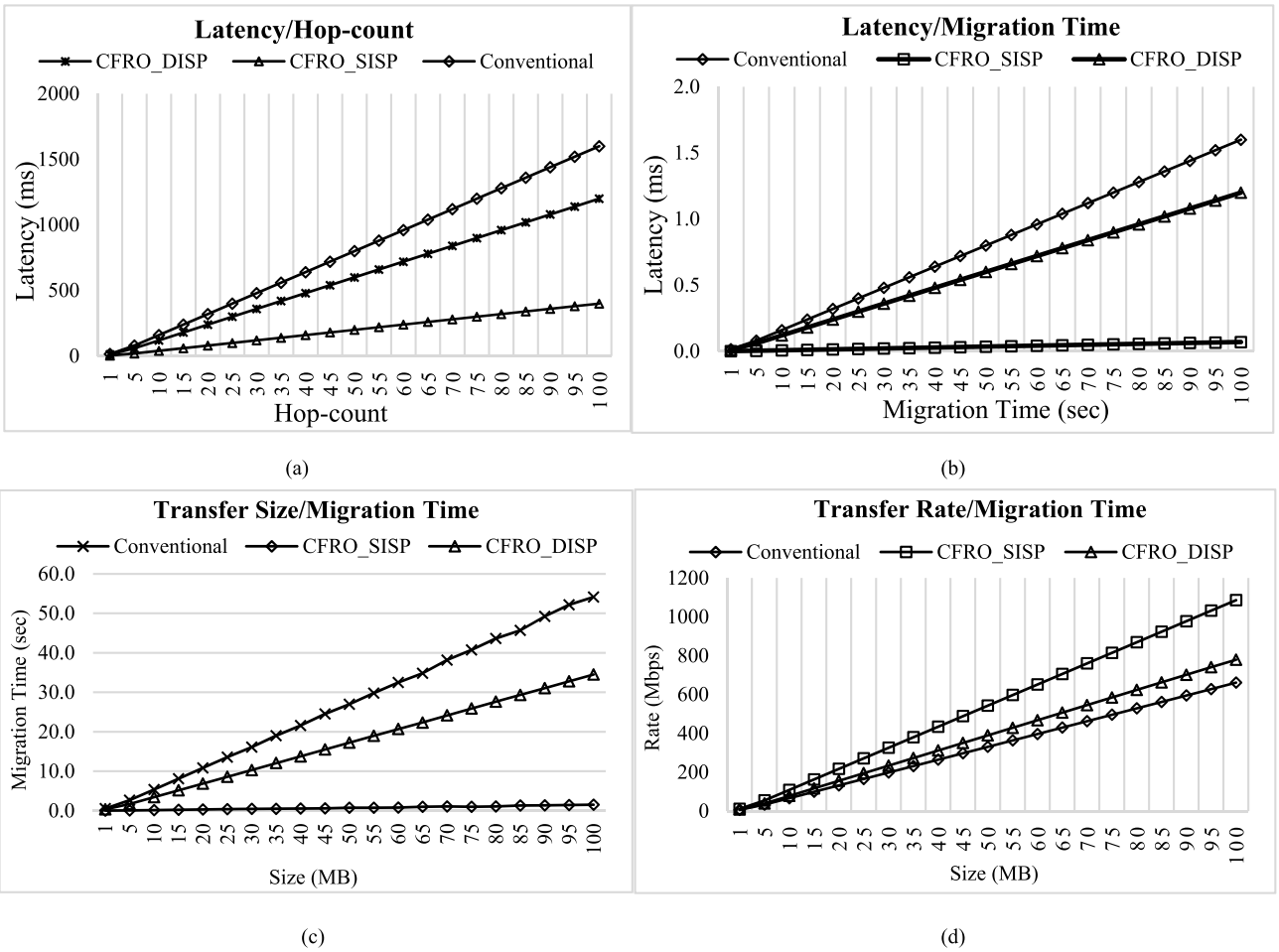

FIGURE 9. Average effect of latency over hop-count and migration time; and average effect of transfer size and rate over migration time.

In the case of different ISPs, the latency is reduced by 36%, and hop-count is reduced by 53%.

2) Migration Time and Throughput: The results shown in Figure 8 indicate an elevated level of migration time for the conventional model due to increased distance, latency and inconsistent throughput as compared to the proposed model. In the case of the same ISP, the migration time is decreased by 39% and in the case of different ISPs, the migration time is decreased by 15% due to reduced distance, hop-count, and latency. The results show an increased throughput for the proposed model as compared to the conventional model. In the case of the same ISP, the throughput is enhanced by 97% and in the case of different ISPs, the throughput is enhanced by 36% due to reduced distance, hop-count, and latency.

3) Average Effect of Latency Over Hop-count and Migration Time: The average effect of latency over hop-count and migration is shown in Figure 9. In the case of the same ISP, the average effect of latency per hop-count is lowered by 67% and the average effect of latency over migration time is lowered by 96%. In the case of different ISPs, the average effect of latency per hop-count is decreased by 25%, and the average effect of latency over migration time is decreased by 36%.

4) Average Effect of Transfer Size and Rate Over Migration Time: The average effect of transfer size and rate over migration time have been presented in Figure 9.

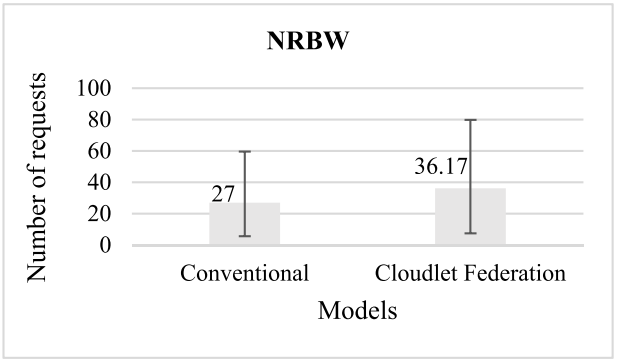

FIGURE 10. Comparison of conventional and CFRO model for the number of requests before worst case.

In the case of the same ISP, the effect of the transfer size over migration time shows a linear trend as maximum throughput is gained due to reduced distance, latency, and hop-count. In the case of different ISPs and conventional models this trend shifts towards exponential. An improvement of 96% is observed in the case of the same and different ISPs, an improvement of 25% is recorded. The results show a 39% improved transfer rate for the same ISP scenario and a 15% improved transfer rate for different ISPs scenario as compared to the conventional model. The reasons for this improvement include stable and maximized throughput, decreased distance, latency, and hop-count.

5) Number of Requests Before Worst Case: This parameter is completely dependent upon available computational resources at a cloudlet. Figure 10 shows an increased number of requests avoiding the worst case to be reached due to federated resources as compared to the conventional model. An improvement of 34% is recorded with a two-member federation. The resource pool increases as more members are added to the federation.

VI. CONCLUSION AND FUTURE DIRECTION

This work plays a pivotal role in the performance improvement of mobile cloud computing. The new model is capable of utilizing untapped resources offered by neighbor cloudlets. This gives an equal opportunity to every service provider for resource and user management as these both are the key elements of a resource federation.

Further, resource-aware resource allocation algorithms for efficient resource management are presented in this study. The experimental results show that the proposed technique brings significant improvement in terms of latency, hop-count, migration time, and number of requests, thus improving the cloudle's performance. The independent nature of the proposed model allows it to be integrated with any third party solution provided by Xen, KVM, VMware, Amazon's Elastic Compute Cloud (EC2), Simple Storage Service (S3), and Microsof's Azure. The potential applications of the proposed cloudlet federation are not only limited to mobile computing but also include the areas of autonomous vehicles, image recognition systems, cyber-physical systems, disaster recovery systems, and missile control systems etc. [44]–[46]. The time-aware federated cloudlet model can further be enhanced by considering the parameters of energy and task awareness.

REFERENCES

[1] W. Liu, W. Gong, W. Du, and C. Zou, "Computation offloading strategy for multi user mobile data streaming applications," in *Proc. 19th Int. Conf. Adv. Commun. Technol. (ICACT)*, 2017, pp. 111–120.
[2] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li, and J. Li, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.
[3] M. H. Ghahramani, M. Zhou, and C. T. Hon, "Toward cloud computing QoS architecture: Analysis of cloud systems and cloud services," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 1, pp. 6–18, Jan. 2017.
[4] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.
[5] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.
[6] Q. Fan and N. Ansari, "On cost aware cloudlet placement for mobile edge computing," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 4, pp. 926–937, Jul. 2019.
[7] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Sep. 2014, pp. 1–8.
[8] P. Zhang, M. Zhou, and G. Fortino, "Security and trust issues in fog computing: A survey," *Future Gener. Comput. Syst.*, vol. 88, pp. 16–27, Nov. 2018.
[9] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proc. 3rd ACM Workshop Mobile Cloud Comput. Services*, 2012, pp. 29–36.
[10] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Comput. Netw.*, vol. 130, pp. 94–120, Jan. 2018.
[11] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.
[12] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Proc. Global Internet Things Summit (GIoTS)*, Jun. 2017, pp. 1–6.
[13] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *J. Netw. Comput. Appl.*, vol. 59, pp. 46–54, Jan. 2016.
[14] D. P. Abreu, K. Velasquez, M. Curado, and E. Monteiro, "A resilient Internet of Things architecture for smart cities," *Ann. Telecommun.*, vol. 72, nos. 1–2, pp. 19–30, Feb. 2017.
[15] M. Z. Nayyer, I. Raza, and S. A. Hussain, "A survey of cloudlet-based Mobile augmentation approaches for resource optimization," *ACM Comput. Surv.*, vol. 51, p. 107, Nov. 2018.
[16] A. Kertesz, "Characterizing cloud federation approaches," in *Cloud Computing*. Cham, Switzerland: Springer, 2014, pp. 277–296.
[17] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. Masoud Sadjadi, and M. Parashar, "Cloud federation in a layered service model," *J. Comput. Syst. Sci.*, vol. 78, no. 5, pp. 1330–1344, Sep. 2012.
[18] R. Buyya, R. Ranjan, and R. N. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Proc. Int. Conf. Algorithms Architectures Parallel Process.*, 2010, pp. 13–31.
[19] E. Carlini, M. Coppola, P. Dazzi, M. Mordacchini, and A. Passarella, "Self-optimising decentralised service placement in heterogeneous cloud federation," in *Proc. IEEE 10th Int. Conf. Self-Adaptive Self-Organizing Syst. (SASO)*, Sep. 2016, pp. 110–119.
[20] C. A. Lee, R. B. Bohn, and M. Michel, "The NIST cloud federation reference architecture 5," NIST Special Publication, Gaithersburg, MD, USA, Tech. Rep., 2020, vol. 500, p. 332.
[21] P. Mell and T. Grance, "The NIST definition of cloud computing," NIST Special Publication, Gaithersburg, MD, USA, Tech. Rep., 2011, vol. 800, p. 145.

[22] M. Z. Nayyera, I. Razab, and S. A. Hussainb, "Revisiting VM performance and optimization challenges for big data," *Adv. Comput.*, vol. 114, pp. 71–112, 2019.
[23] P. Zhang and M. Zhou, "Dynamic cloud task scheduling based on a two-stage strategy," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 772–783, Apr. 2018.
[24] M. Z. Nayyer. *Testbed for Edge Computing*. Accessed: May 2, 2020. [Online]. Available: https://lahore.comsats.edu.pk/Research/Groups/CNRC/QuickLinks.aspx
[25] D. Fesehaye, Y. Gao, K. Nahrstedt, and G. Wang, "Impact of cloudlets on interactive mobile cloud applications," in *Proc. IEEE 16th Int. Enterprise Distrib. Object Comput. Conf.*, Sep. 2012, pp. 123–132.
[26] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Trans. Cloud Comput.*, vol. 5, no. 4, pp. 725–737, Oct. 2017.
[27] S. Bohez, J. D. Turck, T. Verbelen, P. Simoens, and B. Dhoedt, "Mobile, collaborative augmented reality using cloudlets," in *Proc. Int. Conf. MOBILe Wireless MiddleWARE, Operating Syst., Appl.*, Nov. 2013, pp. 45–54.
[28] X. Sun and N. Ansari, "Green cloudlet network: A distributed green mobile cloud network," *IEEE Netw.*, vol. 31, no. 1, pp. 64–70, Jan. 2017.
[29] Y. Jararweh, L. Tawalbeh, F. Ababneh, A. Khreishah, and F. Dosari, "Scalable cloudlet-based mobile computing model," *Procedia Comput. Sci.*, vol. 34, pp. 434–441, 2014.
[30] K. Ha, Y. Abe, Z. Chen, W. Hu, and B. Amos, "Adaptive vm handoff across cloudlets," CMU School Comput. Sci., Pittsburgh, PA, USA, Tech. Rep. CMU-CS-15-113, 2015.
[31] K. Wang, M. Shen, J. Cho, A. Banerjee, J. Van der Merwe, and K. Webb, "Mobiscud: A fast moving personal cloud in the mobile network," in *Proc. 5th Workshop Things Cellular: Oper., Appl. Challenges*, 2015, pp. 19–24.
[32] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervas. Comput.*, vol. 8, no. 4, pp. 14–23, Oct. 2009.
[33] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan, "Just-in-time provisioning for cyber foraging," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2013, pp. 153–166.
[34] M. Chen, Y. Hao, Y. Li, C.-F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: Architecture and service modes," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 18–24, Jun. 2015.
[35] S. Bohez, T. Verbelen, P. Simoens, and B. Dhoedt, "Discrete-event simulation for efficient and stable resource allocation in collaborative mobile cloudlets," *Simul. Model. Pract. Theory*, vol. 50, pp. 109–129, Jan. 2015.
[36] H. Flores, R. Sharma, D. Ferreira, V. Kostakos, J. Manner, S. Tarkoma, P. Hui, and Y. Li, "Social-aware hybrid mobile offloading," *Pervas. Mobile Comput.*, vol. 36, pp. 25–43, Apr. 2017.
[37] Y. Wu and L. Ying, "A cloudlet-based multi-lateral resource exchange framework for mobile users," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 927–935.
[38] K. A. Khan, Q. Wang, C. Grecos, C. Luo, and X. Wang, "MeshCloud: Integrated cloudlet and wireless mesh network for real-time applications," in *Proc. IEEE 20th Int. Conf. Electron., Circuits, Syst. (ICECS)*, Dec. 2013, pp. 317–320.
[39] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "A context sensitive offloading scheme for mobile cloud computing service," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 869–876.
[40] X. Guo, L. Liu, Z. Chang, and T. Ristaniemi, "Data offloading and task allocation for cloudlet-assisted ad hoc mobile clouds," *Wireless Netw.*, vol. 24, no. 1, pp. 79–88, 2018.
[41] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1285–1293.
[42] H. Mazouzi, N. Achir, and K. Boussetta, "Dm2-ecop: An efficient computation offloading policy for multi-user multi-cloudlet mobile edge computing environment," *ACM Trans. Internet Technol. (TOIT)*, vol. 19, pp. 1–24, 2019.
[43] I. Adan and J. Resing, *Queueing Theory*, ed. Eindhoven, The Netherlands: Eindhoven Univ. Technology, 2002.
[44] L. Chen, X. Hu, W. Tian, H. Wang, D. Cao, and F.-Y. Wang, "Parallel planning: A new motion planning framework for autonomous driving," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 1, pp. 236–246, Jan. 2019.
[45] G. Bhatnagar and Q. J. Wu, "A fractal dimension based framework for night vision fusion," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 1, pp. 220–227, Jan. 2019.
[46] S. M. Rahman, "Cyber-physical-social system between a humanoid robot and a virtual human through a shared platform for adaptive agent ecology," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 190–203, Jan. 2018.

MUHAMMAD ZIAD NAYYER received the M.S. degree in computer science from the Government College University (GCU), Lahore, Pakistan, in 2011, and the Ph.D. degree in computer science from COMSATS University Islamabad–Lahore. He is currently serving as an Assistant Professor with the Department of Computer Science, GIFT University, Gujranwala, Pakistan. He is an Active Member of Advanced Communication Networks Lab. He has numerous publications on his account, including impact factor journal publications and book chapters. His research interests include cloud computing, VM migration, mobile cloud computing, cloud federation, mobile edge computing, fog computing, and cloudlet computing.

IMRAN RAZA (Member, IEEE) received the B.S. (CS) and M.Phil. degrees in computer science from Pakistan. He has been working as an Assistant Professor with the Department of Computer Science, COMSATS University Islamabad–Lahore, since 2003. He has authored and coauthored more than 40 journal and conference papers. He has been actively involved in simulating CERN O2/FLP upgrades. He has supervised and co-supervised many funded projects related to ICT in Healthcare. His research interests include cloud computing, mobile edge computing, SDN, NFV, wireless sensor networks, MANETS, QoS issue in networks, and routing protocols. He has been member of ACM.

SYED ASAD HUSSAIN received the master's degree from Cardiff University, U.K., and the Ph.D. degree from Queen's University Belfast, U.K. He was the Head of the Computer Science Department, COMSATS University Islamabad–Lahore, Pakistan, from August 2008 to August 2017. He has been serving as the Dean of Faculty of Information Sciences and Technology, since 2015. He is currently leading communications and networks research with COMSATS University Islamabad–Lahore. He is supervising Ph.D. students at CUI and split-site Ph.D. students at Lancaster University, U.K., in the fields of cloud computing and cybersecurity. He was funded for his Ph.D. by Nortel Networks UK Ltd., at Queen's University Belfast. He has taught at Queen's University Belfast, the Lahore University of Management Sciences (LUMS), and the University of the Punjab. He was awarded prestigious endeavour research fellowship for his post doctorate at The University of Sydney, Australia, in 2010, where he conducted research on VANETs. He regularly reviews IEEE, IET, and ACM journal articles.