Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Highlights

Future Generation Computer Systems xxx (xxxx) xxx

An SDN based framework for maximizing throughput and balanced load distribution in a Cloudlet network

Shirzad Shahryari, Seyed-Amin Hosseini-Seno[*], Farzad Tashtarian

- A novel framework was designed for managing the resources pool of a Cloudlet network for the shared use of all the users based on the SDN approach.
- An MILP model was proposed for the problem of admitting and balancing the input load based on available resources with the goal of maximizing throughput.
- Because the MILP model proves to be NP-hard and has a high complexity in a large-scale network, an LP-relaxation model was proposed.

Future Generation Computer Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An SDN based framework for maximizing throughput and balanced load distribution in a Cloudlet network

Shirzad Shahryari [a], Seyed-Amin Hosseini-Seno [a,*], Farzad Tashtarian [b]

[a] Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[b] Department of Computer Engineering, Islamic Azad University, Mashhad, Iran

ARTICLE INFO

Article history:
Received 23 August 2019
Received in revised form 17 March 2020
Accepted 6 April 2020
Available online xxxx

Keywords:
SDN
Mobile Cloud Computing (MCC)
Cloudlet
Load balancing
Mobile Edge computing (MEC)

ABSTRACT

Although mobile devices have experienced voluminous proliferation throughout the last decade, there are limited resources in terms of their portable size. Such limitations could be mitigated by remote execution of the computation-intensive tasks to the cloud. By creating a cluster of servers (a.k.a. "Cloudlets") to the network edge and close to the mobile devices, task offloading could be performed with a more acceptable delay in comparison with a cloud-based solution. Nevertheless, once the user requests mount, the resource constraints in a Cloudlet will lead to resource shortages. However, this challenge can be obviated using a network of Cloudlets for sharing their resources. This paper proposes a novel framework to optimally manage the resources and balance an equitable load across a network of Cloudlets via software-defined networking (SDN) techniques. To achieve this, firstly, the problem is considered as a mixed-integer linear programming (MILP) optimization model in order to balance the distribution of independent tasks offloaded from the mobile devices along with optimal use of resources. The MILP model guarantees meeting the tasks' deadlines and maximizes overall system throughput. Secondly, by showing that the addressed problem is NP-hard, an LP-relaxation model is proposed to enable the SDN controller on a large-scale network. Finally, we conduct experiments by emulating the proposed framework in Mininet-WiFi, with the Floodlight usage as the SDN controller. The simulation results indicate that the proposed architecture can achieve a significant throughput maximization of a system, which satisfactorily performs load balancing, and offers adequate proof, as well.

1. Introduction

In recent years, with the rapid progress of cloud and mobile computing technologies, the pervasive increase of mobile devices and the popularity of various kinds of mobile applications, mobile traffic data has been rising at an unprecedented rate. Based on several scientific reports including the Cisco visual networking index (Cisco VNI^{TM}), there will be a growth in the number of mobile devices and connections from 8 billion in 2015 to 11.6 billion in 2020 [1]. Furthermore, there will be an increase in mobile traffic from 7 exabytes per month in 2015 to 49 exabytes in 2020. Owing to this rapid growth in the demand for mobile applications, mobile network operators (MNOs) have felt a conspicuous responsibility to deliver quality service for the new computation-intensive applications. Bearing this in mind, it is asserted that, by now, creating a new framework to maintain both scalability and quality of service (QoS) is undeniable [2].

Since mobile devices have limited resources in terms of memory size, processing power, battery lifetime, and communication bandwidth, they are inappropriate for running the applications requiring numerous resources. To eradicate such issues, mobile cloud computing (MCC) has proposed to cater to this need. it is a model for offloading mobile device tasks to a cloud (known as computation offloading) with high resources resulting in the mobile device resource constraints could be mitigated.

Offloading the tasks to the cloud is not appropriate for a variety of mobile applications, due to diverse reasons such as the long average access delay between users and remote clouds, high power consumption, and the lack of local user information [3]. To unravel these issues, it is suggested to use emerging technologies such as offloading mobile network traffic to the complementary networks (e.g., Wi-Fi and satellite-terrestrial networks [4]) and edge cloud computing (ECC)). ECC technology has been proposed with the MEC, Cloudlet, and fog approaches.

MEC is introduced by ETSI in [5] which is designed to bring resources closer to the radio access network (RAN) in the 4G and 5G cellular networks. Therefore, this environment is specified by high bandwidth as well as low latency that can be exploited by applications.

* Corresponding author.
E-mail addresses: sh.shahryari@mail.um.ac.ir (S. Shahryari), hosseini@um.ac.ir (S.-A. Hosseini-Seno), f.tashtarian@mshdiau.ac.ir (F. Tashtarian).

https://doi.org/10.1016/j.future.2020.04.009
0167-739X/© 2020 Elsevier B.V. All rights reserved.

The term “Cloudlet” has devised by M. Satyanarayanan at Carnegie Mellon University [6]. The Cloudlets are designed to support applications that are resource hungry and interactive for mobile devices to achieve the required quality of experience (QoE). This assists to reduce communication latency and to do faster execution for applications intending to perform resource-intensive tasks. Cloudlets can be considered as a local data center to enable localized cloud services with less end to end (E2E) latency and the high bandwidth available to multiple users, simultaneously. Thereupon, the user’s equipment (UE) can access more powerful computing resources and large storage resources with less E2E latency.

Fog computing to meet the demands from different segments of the Internet of things (IoT) is a concept proposed by Cisco in 2011 [7]. It introduces distributed computing infrastructures where the application services and computing resources are distributed on efficient places at any point along the continuum from the data source to the classical cloud.

By leveraging the Cloudlet approach, the mobile applications which require high computational power, memory, storage and energy can be executed in the powerful servers that are associated with base stations (BSs) and located near the mobile devices. Since the resources in a Cloudlets are not as many as those in a cloud, they may run out instantly if carelessly allocated. The Cloudlet overload will particularly increase many requests are executed, simultaneously. For optimal use of the Cloudlet resources and prevention of the Cloudlet overload, it is reasonable and cost-effective to connect them to each other through a mobile core network (MCN) [6].

To shorten the response times of the offloaded tasks, MNOs are required to determine a method that allocated the user tasks to different Cloudlets so that the loads among the Cloudlets will be well-balanced. To untangle this problem, the user requests can be allocated to the closest Cloudlet in order to reduce access delay. This solution, however, has been reported to be inadequate in the mobile network setting [8]. The workload of each Cloudlet will be subject to change, particularly when the number of users of the network rises. If a cloudlet is suddenly overwhelmed with user requests, there will be a dramatic increase in the response times of the tasks in that Cloudlet, leading to lags in the user applications, a degradation of the QoE from mobile users’ point of view, and a decline in the network throughput, accordingly [9].

In response to the dynamic demands of the users’ request, the present study deals with the load balancing problem among the Cloudlets by employing an efficient model to allocate users’ request to various Cloudlets. The current work associates hard deadlines to offloaded tasks in order to ensure the QoS for mobile applications. After the arrival of an offloaded task, the Cloudlet to which this task is assigned will be determined. The task will be finally executed in the assigned Cloudlet according to its associated deadline.

To balance the load among a few Cloudlets via a centralized approach, the use of hardware balancer is inappropriate for various reasons, including high costs, complex management, and hardware constraints. Hence, softwarization of the load balancer is indispensable. Thus, SDN is reasonably employable. By employing the SDN controller, alongside predesigned APIs (e.g., the OpenFlow protocol (OF) [10]), the entire network and its elements may be controlled and programmed as a unified network. The architectures of the SDN and the OF protocol allow separating the data and control plane, which contributes to the control of the network, effective routing, resource management, plus rendering the network even more smartly. Furthermore, separating the network infrastructure from the applications would be possible [11].

To the best of our knowledge, the joint consideration of maximizing the system throughput and balanced load distribution in a Cloudlet network with the optimal use of resources based on the SDN approach has not yet been addressed in the existing literature to this date. The contribution of this paper could be summarized as follows:

1. At first, we design a new framework for a Cloudlet network on the edge of mobile network based on the SDN technology.
2. Afterward, this work defines the problem of a balanced distribution of incoming requests among the Cloudlet network by taking into account both the communications and computing latency, available resources, and a deadline-constrained task. The purpose of this problem is to maximize the system throughput gain and optimal resource utilization.
3. Then, the addressed problem in the multi-Cloudlet environment is formulated as a multi-objective MILP model. This approach considers the users’ request requirements and available resources as constraints. Because the problem proves to be NP-hard and has a high complexity in a large-scale network, an LP-relaxation model has been proposed.
4. At the end, several experiments have conducted via simulations in order to evaluate the performance of the proposed model. Experimental results attest that the proposed model is more efficient, in comparison with baseline and heuristic algorithms, such as the First-Come-First-Service strategy (FCFS), the Closest-Cloudlet-First Assignment approach (CCFA), Online-Batch (OB) [8], and Cloudlet Load Balancing Algorithm (CLBA) [12].

The rest of the paper is organized as follows. Section 2 provides an overview of the related works. Section 3 presents the problem description and motivating example. Section 4 discusses the proposed framework and system model. The performance evaluation of the proposed models is explained in Section 5. Finally, concluding remarks and future perspectives are described in Section 6.

2. Background and related works

The extensive reviews on mobile edge computing can be found in [3]. Principally, existing studies for Cloudlet load balancing can be classify into two groups, i.e., computation offloading optimization and optimal Cloudlet placement. There have been some studies focusing on the user’s task offloading to single Cloudlets or multi-Cloudlet environment rather than the remote clouds. The mobile users’ task offloading on the Cloudlets can be accomplished with a number of goals, include optimizing device energy [13], application latency [14], increased QoS as well as QoE [15], Cloudlets workload [12,16], optimal allocation of the available resources [17,18], and increase the system throughput [8].

The authors in [14] have studied the tasks offloading to a Cloudlet in a VM-based manner, where the mobile device takes a compressed VM image from the application and transfers it to the Cloudlet. Therefore, it is possible for the mobile devices to be able to remotely execute their tasks in VM in the Cloudlet. This technique reduces the power consumption of the mobile devices to better perform applications that require high computation.

Several tests conducted by various research projects report that the Cloudlets when compared with the remote clouds, can improve the response time and energy consumption of the mobile devices by 51% and 42%, respectively [13]. The task offloading between the cloud and the Cloudlet at the same time has been investigated with respect to energy consumption and QoS in [15].

It is believed that there are many similarities between the placement of Cloudlets and edge servers [12,16,19]. The optimal placement of Cloudlets in wireless metropolitan area networks (WMAN) and the design of an algorithm to allocate the users request to the Cloudlets have been studied in [16]. They further proposed an algorithm for load balancing among multiple Cloudlets [19]. In this research, the priority of offloading the tasks is the Cloudlet, which has a wireless connection to the mobile device via a hop (as a home Cloudlet). The home Cloudlet may sometimes face increasing loads which lengthens the delay in queuing for task scheduling and may cause the task to fail due to the required deadline. If the home Cloudlet encounters an increasing load, it sends the overload to the other Cloudlets experiencing low load. The main drawback of this study is the time complexity of its algorithm (because of the repeated sorting process), especially with the increasing number of Cloudlets and the lack of attention to user mobility.

Moreover, [19] and [12] thoroughly have examined the way to optimally deploy Cloudlets distributed in an urban public network so that the user requests will be responded in the shortest possible time and according to application requirements. In these studies, K Cloudlets have been located in potential locations in a wireless urban network for the purpose of minimizing the intermediate latency of the access to the Cloudlet by mobile users and the Cloudlets serving their requests. Approximate algorithms for solving these problems are provided when all the Cloudlets have the same computational capacity. The most critical shortcomings of these studies are the complexity of the proposed algorithms, particularly if the user requests are unbalanced in terms of the resources required. Other demerits include considering the Cloudlet resources as homogeneous and the possibility of providing all the services in all the Cloudlets.

However, these studies have attempted to manage resources in the large-scale networks using local information at each node. Hence, this method is not suitable for WAN networks where Cloudlets are distributed over a wide geographic area with high latency, low bandwidth, and unreliable links [20].

Based on its long-term evolution (LTE) cellular network, [20] proposes an architecture for a Cloudlet network as a means of providing proper computational resources for UEs while simultaneously maintaining minimal E2E latency. In this work, a VM is created in the home Cloudlet for each user. Due to the user's mobility, the connection point (i.e., BS) changes if the user moves from one cell to another, while the user's VM is located in the Cloudlet connected to the previous BS. In this case, the delay between the user's VM and the UE increases, which may not be tolerable for running application in VM. In order to reduce this delay, the user's VM, should be migrated to the new home Cloudlet. It should be noted, however, that the VM live migration with its services, through the wide area network (WAN) is costly and the authors have tried to make optimal use of the network resources by balancing between E2E delay and the cost of live migration.

Many studies have focused on allocating computing workloads among edge clouds without considering the traffic load balancing in mobile networks [21]. A workload assignment algorithm in a hierarchical edge cloud network in order to optimize the response time of all tasks has been proposed in [22]. The algorithm assigns tasks among different tiers of fog nodes and allocates the computing resources of each fog node for their assigned tasks.

In a multi-Cloudlet environment, optimal resource management for offloading is done within two phases; selecting the most appropriate Cloudlet and optimizing resource allocation for each task. A two-stage optimization strategy has been introduced in [17]. They have considered a Cloudlet selection and a resource allocation model based on MILP to obtain the Cloudlet for mobile users by optimizing latency and mean reward. Second, a resource allocation model based on MILP is presented to allocate resources in the selected Cloudlet by optimizing reward and mean resource usage. Due to the highly dynamic traffic loads of mobile devices and a large number of access points in the IoT network, it is difficult to efficiently deploy Cloudlets. An optimal Cloudlet selection strategy has been proposed in [18] to reduce the power consumption and latency in multi-Cloudlet environment.

In contrast to the existing studies, to efficient resources management, our research utilizes the SDN approach as a promising technology for simple, inexpensive, scalable, and flexible management. By centralized network planning and a comprehensive look at the environment, the researchers will address the challenge of proper Cloudlet selection and optimal resource allocation in multi-Cloudlet environment for independent input tasks.

3. Problem description and motivating example

Nowadays, one of the main concerns of MNOs is providing high-quality services to the most demanding users with applications require huge amounts of computing resources along with optimizing the use of network's valuable resources. To this end, to increase QoS and user satisfaction and operators' revenue in mobile network implementation, two significant issues should be considered: maximizing acceptance of requests and optimizing the use of resources.

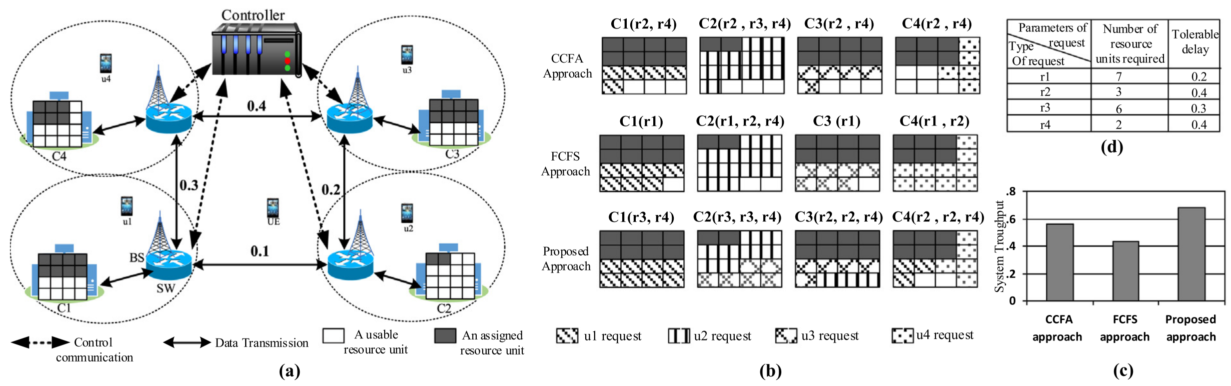
With the emergence of Cloudlet-based ECC technology, it has provided MNOs with the opportunity to provide cloud services with low latency and high bandwidth for mobile applications requiring high resources. Nevertheless, resource-efficient management to appropriately allocate each request to a Cloudlet is a challenge due to the dynamically changing users' demand, user mobility, geographically distributed Cloudlets, and diverse applications in terms of the required resources. These factors, alone, influence the load of the Cloudlets and the amount of resources they use. Thus, the Cloudlet resources can be quickly overloaded in one Cloudlet, while another Cloudlet in another geographical area is under-loaded.

However, overload in a Cloudlet increases the response time of the task and reduces service rate. In the users' point of view, it reduces QoS as well as QoE and in the operator's view, it reduces revenue. To eradicate such issues, sharing resources across all Cloudlets is proposed by balancing requests between them with a dynamic approach.

Before describing the proposed optimization model, let us consider an example illustrated in Fig. 1(a). It is assumed that a simple topology with four cells connected to the SDN controller via an OpenFlow network. Each cell contains a BS to get the requests from the mobile users and a Cloudlet for the requested service. The transmission delay of any link and available resources to each Cloudlet are illustrated in Fig. 1(a). In the *i*th time slot, the SDN controller receives the four requests r1, r2, r3, and r4 from each user u1, u2, u3, and u4. The characteristics of each request are presented in Fig. 1(c).

Now, to assign these requests to the Cloudlets, we have been defined three approaches as follows:
Scenario 1: assign each request to the nearest Cloudlet (i.e., CCFA approach)
Scenario 2: assign requests based on arrival time the controller (i.e., FCFS approach)
Scenario 3: use all Cloudlets resources to allocate maximum requests regardless of proximity or reception time (our proposed approach).

Fig. 1(b) illustrates the allocation of requests to the Cloudlets in these three approaches. Now, to evaluate the efficiency of the above three scenarios, the total system throughput has been



Parameters of request / Type Of request	Number of resource units required	Tolerable delay
r1	7	0.2
r2	3	0.4
r3	6	0.3
r4	2	0.4

Fig. 1. Scenario of sending the 4 requests from each user to the SDN controller at time slot *i* (a), allocating user requests to Cloudlets in the CCFA, FCFS and our proposed approaches (b), the characteristics of each request (c), and the throughput value of each approach (d).

defined as the number of requests accepted by all Cloudlets to the total number of requests.

In scenario 1, for the set of tasks received from each BS, each task is assigned to the nearest Cloudlet according to the resources available. In this allocation, the goal is to accept the most tasks and make maximum use the resources of each Cloudlet. For instance, the c1 has accepted r2 and r4.

In scenario 2, for the set of the received tasks, each task is allocated based on the order of receipt, the required resources and the available resources in each Cloudlet. For example, the c1 has accepted r1.

In scenario 3, it attempts to accommodate most tasks that require fewer resources in order to maximize throughput. Consequently, tasks requiring high resources will not be automatically accepted if the required resources are not available.

As illustrated in Fig. 1(d), by distributing incoming requests among the shared resources of Cloudlets, overall system throughput increases, resulting in more operator revenue and more user satisfaction.

In a network of many users and Cloudlets, in order to select the most appropriate Cloudlet for each task and allocate the resources to it, the available resources of all Cloudlets should be considered. However, the balanced distribution of requests between Cloudlets with the aim of maximizing system throughput requires an appropriate architecture with a comprehensive overview of all network resources that we have thoroughly studied in this research.

In this paper, to deal with the above challenges, we proposes a dynamic load balancing scheme based on the SDN approach for the multi-Cloudlet environment to maximize the throughput of users' requests. This work integrates the communications and computing latency with the available resources to achieve effective load balancing in the Cloudlet network.

3.1. Classification of load balancing approaches

The Cloudlet network provides on-demand access to a shared pool of resources at the edge of mobile networks, that requires a great amount of control and management of the user's request and resources. To manage user requests for the available resources, an appropriate load balancer is required to meet the requirements of this environment and allocate the requested tasks to Cloudlets. The Cloudlet observes a high variation in user requests due to user mobility that require the dynamic environment to execute the tasks. Balanced load distribution in a dynamic environment requires an agile, flexible and scalable balancer. Dynamic load balancing techniques are flexible, which leads to improvement in system performance. The dynamic load-balancing schemes can be categorized as follows:

(1) Distributed methods: in distributed methods [12], all Cloudlets engage in load distribution and each Cloudlet must have system status information in real-time. The extra overhead to exchange system status information between Cloudlets and the convergence time between them, especially for WAN networks where Cloudlets are distributed over a wide geographic area with the high latency, low bandwidth, and unreliable links, make them inefficient for use in a large-scale network.

(2) Non-distributed methods: in non-distributed methods, a single controller or some controllers make the decision for load distribution. Non-distributed approaches with all the advantages over distributed methods (e.g., flexible, inexpensive and simple management), however, create limitations such as single point failure and suffer from poor scalability due to the central controller. Non-distributed techniques can be centralized or semi-distributed in manner. In centralized techniques, a single node performs all load distribution activities and is responsible for load balancing [23]. In the semi-distributed technique, clusters are organized in Cloudlets and each cluster works as a centralized technique [24].

The SDN technology as a Non-distributed technique is able to provide flexible, inexpensive and simple management with a comprehensive look at the status of the system at any time [25]. In our proposed framework, without loss of generality, and in order to simplify the experiment, we assume that the whole system is managed by a single SDN controller. Meanwhile, our proposed framework can be further extended to multi-controllers scenario. Compared with a single controller scenario, the major additional problem is the interplay and harmony between various controllers.

There are already many related works aim to solve this problem, in which using a central coordinator for controllers is considered to be an excellent choice. [26] has been used a central coordinator for harmony among controllers and cloud management systems. Such techniques can also be applied to our approach to change the control plane of SDN network into a two-layer structure, where all the controllers are coordinated by a global controller. A global controller can be used for information interplay and synchronization between the controllers. In this paper, all the models have been implemented and analyzed in a single controller scenario.

4. The details of the proposed framework

In this section, we investigate the details of the proposed framework. First, let us introduce the schematic architectural model of the proposed framework as shown in Fig. 2. In this architecture, $\mathcal{U}$ as a group of mobile users that connects wirelessly to the set of BSs, $\mathcal{A}$. The set of BSs associated with the set of proxy

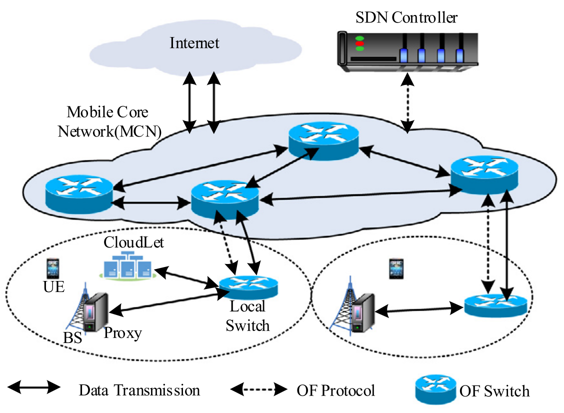


Fig. 2. The Communication topology of the proposed framework.

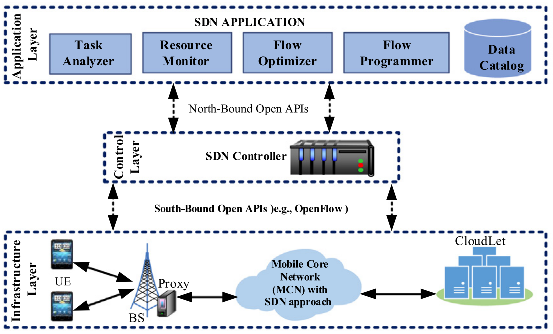


Fig. 3. Three-layer model of the proposed framework.

Table 1
Notations.

Parameter	Definition
$\mathcal{C}$	A set of Cloudlets
$\mathcal{SW}$	A set of OF switches
$\mathcal{S}$	A set of services to be provided by $\mathcal{C}$
$\mathcal{A}$	A set of base stations(BSs)
$\mathcal{U}$	A set of users connected to the network
$\mathcal{T}$	A set of tasks requested by users
$\mathcal{P}$	A set of proxies
$\mathcal{V}$	A set of $\mathcal{SW}$, $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{C}$
$\mathcal{E}, e_{i,j}$	The two-dimensional array $\mathcal{E}$, where $e_{i,j}$,represents the edges of graph $\mathcal{G}$ so that $e_{i,j} = 1$, if there is a direct link between two entities $i, j \in \mathcal{V}$; otherwise, $e_{i,j} = 0$
$\rho_{u,i}$	Indicator representing the connection of user u to BS i
$\theta_{t,u}$	Duration of task t for user u which should be sent from a proxy to the Cloudlet
$B, b_{i,j}$	The two-dimensional array B, where $b_{i,j}$ represents the available bandwidth between nodes $i, j \in \mathcal{V}$
s_t^c	Parameter indicating if the service requested for task t is provided by the Cloudlet c
$l_{t,u}$	Data size the task t for user u
$\pi_{t,u}$	Task resize Coefficient after servicing by the selected Cloudlet
χ_i	Percentage of the available resources in each Cloudlet i at each time slot τ
ξ_t	Percentage of the required resources of a Cloudlet for running task t
Variable	**Definition**
$O_{t,u}^c$	A binary variable that shows whether task t from the user u is assigned to the Cloudlet c
$w_{i,j}^{t,u,c}$	The raw traffic of task t for user u between node i and j to the Cloudlet c
$\bar{w}_{i,j}^{t,u,c}$	The processed traffic of task t for user u between node i and j from the Cloudlet c
R	The maximum bandwidth available on the links after assigning tasks in each time slot τ
M	The maximum incoming traffic rate to the Cloudlet

has been denoted by $\mathcal{P}$. Each user who requests different types of services denoted by $\mathcal{S}$ to the set of Cloudlets, $\mathcal{C}$, each capable of serving different types of requests. As well as, it is assumed that a request is considered as independent and executable tasks in each Cloudlet and an MNO has setup Cloudlets at fixed locations on the edge of the mobile network and all the Cloudlets are connected to each other through the SDN based MCN (i.e., backbone network). Moreover, they use this connectivity to delegate the execution of requests to each other. For convenience, we assumed that the network operation is divided into a discrete set of time slots $\delta = \{1, 2, \ldots, \tau\}$, with equal duration $\theta*$. In addition, it is assumed that the system has no knowledge of the future generation of requests and arrival rates.

Let us define $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ as an undirected graph, so that $\mathcal{V}$ stands for a collection of all the OF switches, $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{C}$, while $\mathcal{E}$ represents the set of edges of $\mathcal{G}$, so that $e_{i,j} = 1$ if there is a direct link between the two entities $i, j \in \mathcal{V}$; otherwise, $e_{i,j} = 0$. Additionally, each Cloudlet $c_i \in \mathcal{C}$ has χ_i computing capacity at the beginning of time slot τ. Table 1 tabularizes the main notations used in the proposed models.

To clarify and investigate the operation of the proposed approach, a three-layered architectural model consisting of the infrastructure, control, and application layers, along with the elements of each layer is illustrated in Fig. 3.

In the infrastructure layer, the main elements are the OF switches, BSs, proxies, and Cloudlets. The SDN controller in the control layer is in charge of serving the tasks based on the user requests by implementing the modules defined in the application layer.

Given the requests received from the proxies in each time slot τ and requirements of each request (e.g., the complete execution time and the required resources) and the resources available in the Cloudlets and network (e.g., the link bandwidth), the proposed model will determine an optimal solution for offloading each request to the most appropriate Cloudlet to meet its needs. Fig. 4 presents the exchanged messages between the elements for offloading a task to an appropriate Cloudlet in a time slot τ, taking into account the deadline to complete execution. These messages are:

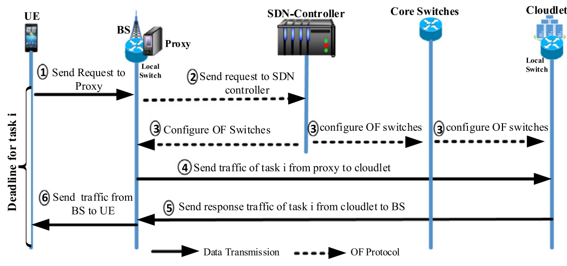


Fig. 4. The exchanged messages among the system elements for the proper offloading of task i to a Cloudlet.

1. Request For offloading task i: sending the task i from the UE to the proxy for offloading to the Cloudlets.

2. Send request to an SDN controller: sending the first packet of task i from the proxy to the SDN controller (as a Packet-In) to select the appropriate Cloudlet.

3. Configure the OF switches: sending rules to the OF switches to configure them (as a Packet-Out).

4. Send traffic of task i: sending traffic of task i from the proxy to the selected Cloudlet from the configured switches path.

5. Send response traffic of task i: sending the traffic for the response of task i from the Cloudlet to the BS.

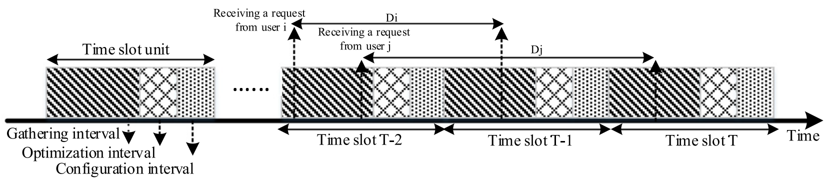


Fig. 5. The structure of time slots in the proposed framework.

6. Send response traffic of task i: sending the traffic for the response of task i from the BS to the UE.

In the proposed layer-based architecture model, the application layer consists of the main following modules: task analyzer, resource monitor, data catalog, flow optimizer, and flow programmer module.

Task Analyzer Module (TAM): This module is charged with the user-requested tasks from the proxies at any time slot τ. In reality, each time slot τ is divided into three unequal intervals, namely gathering, optimization, and configuration (see Fig. 5).

In the gathering interval, the TAM module buffers the received tasks and provides some of the initial processing to prepare them as input parameters to the flow optimizer module (FOM) for the next interval. In the preprocessing stage, the TAM module extracts some of the main values (i.e., the key parameters) from the user requests and sends them to the FOM module as input parameters. Based on the extracted parameters, the TAM module creates $\mathcal{A}$, $\mathcal{U}$, and $\mathcal{T}$ sets as well as $\rho_{u,i}$. The $\rho_{u,i}$ parameter represents whether the user $u \in \mathcal{U}$ is connected to the BS $i \in \mathcal{A}$ at the time slot τ or not (1 and 0, respectively).

In the optimization interval, the proposed model is performed and specified: whether accepted or not, the selected Cloudlet for each accepted task, the traffic transfer path of each task and the transmission data rate for each task at each link. In the configuration interval, the flow programmer module (FPM) configures the OF switches using the OF protocol based on the output of the FOM module. As depicted in Fig. 5, the two users who submit their requests in gathering interval at the time slot τ-2 should both wait for the optimization interval in order to determine an optimal solution for distributing the tasks among the Cloudlets.

Resource Monitor Module (RMM): Using the RESTful APIs, the RMM module reads the link bandwidth between $\mathcal{V}$ entities in gathering interval at each time slot with the SDN controller. The measured values are stored in two-dimensional array B where $b_{i,j}$ represents the available bandwidth between entities $i, j \in \mathcal{V}$. The communication topology between the switches, BSs, and Cloudlets can be easily extracted through array B. The module also calculates the available resources in each Cloudlet according to the resources allocated and released in the previous time slot.

Transmission delay refers to the time spent transmitting data between the user and the Cloudlet, i.e. when the user sends the task to the Cloudlet and the Cloudlet transmits the output back to the user after the task is executed. This time can be expressed as follows:

$$\theta_{t,u} = \frac{D_t - \lambda_t^i}{2}.$$

where λ_t^i is the service providing time of t-type task in the Cloudlet i (for each task type, this time is assumed to be known and constant) and D_t is the completion time of task t (the deadline time).

When a task in a Cloudlet requires more than a one-time slot to run, the assigned resources to that task are not released for multiple time slots. The number of time slots for each task shown by n_t is as follows:

$$n_t = \lfloor \frac{D_t}{\theta^*} \rfloor.$$

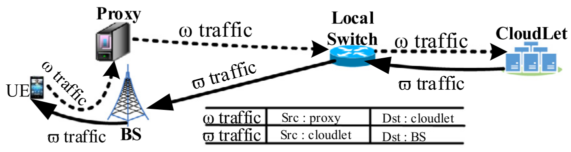


Fig. 6. The types of traffic in the proposed models.

where θ^* is the duration of each time slot.

In general, this module is responsible for creating graph $\mathcal{G}$ to calculate the available bandwidth between $\mathcal{V}$ nodes.

Data Catalog Module (DCM): This module provides critical meta-data about the Cloudlets including a list of services provided by all Cloudlets ($\mathcal{S}$). Afterward, it characterizes parameter s_t^c for task t in Cloudlet c. Furthermore, the residual resource available on each cloudlet c (χ_c) is updated at the end of each time slot for using the optimization problem in the next time slot. The amount of the required resources (ξ_t) and number of required time slots to complete executing (n_t) for t-type tasks are the other parameters in the database.

In the present research, it is assumed that tasks of the same type have the same parameters. These parameters are stored in this module, which may be used at any time slot by the FOM module.

Flow Optimizer Module (FOM): The FOM module introduces a MILP model for optimal resource management, which features optimal data paths for delivering user-requested tasks to the Cloudlets by maximizing the system throughput and fairness in the distribution of the loads among the Cloudlets. Hence, to find the optimal solution for such a problem, a number of constraints must be satisfied. Some of the input parameters in this problem are statically provided by the DCM module and a part of the other parameters is dynamically provided by the TAM and RMM modules.

Let the binary variable $O_{t,u}^i$ determine whether the problem constraints are provided for offloading task t for user u in Cloudlet i, then $O_{t,u}^i = 1$ else, $O_{t,u}^i = 0$. For each $s \in \mathcal{S}$, the s_t^i parameter is used to represent whether Cloudlet i can serve a t-type task or not (1 and 0 respectively). The value of this parameter for each service is statically stored in the DCM module by the operator. The present work defines the traffic from the proxy to the Cloudlet as a raw traffic (i.e., w) while we consider the response traffic of the Cloudlet to the BS as the processed traffic (i.e., $\bar{w}$)(see Fig. 6). As depicted in Fig. 6, for w traffic, the proxy is the source for receiving the requests from the users and the Cloudlet is the destination. Whereas, for $\bar{w}$ traffic, the Cloudlet is the source and the BS is the destination for delivery to the users.

The goal of the model is to assign each task from each user to exactly one Cloudlet of set $\mathcal{C}$ at a time slot τ, which means that the following condition must be satisfied:

$$\sum_{i \in \mathcal{C}} O_{t,u}^i \leq 1 \qquad \forall t \in \mathcal{T}, u \in \mathcal{U} \qquad (1)$$

However, depending on the different services required by the tasks and the operator's policies, each Cloudlet is configured for a number of services. Therefore, a user request must be accepted in the Cloudlet where the required service for that task is in that Cloudlet, which is ensured by the following constraint:

$$O^i_{t,u} \le s^i_t \qquad \forall i \in \mathcal{C}, t \in \mathcal{T}, u \in \mathcal{U} \tag{2}$$

In each switch, the total raw incoming traffic from various users must be equal to the raw outgoing traffic and the total processed incoming traffic must be equal to the total processed outgoing traffic. To do this, the following two constraints must be satisfied:

$$\sum_{j \in \mathcal{SW}} e_{i,j} w^{t,u,c}_{i,j} - \sum_{j \in \mathcal{SW}} e_{j,i} w^{t,u,c}_{j,i} = 0 \quad \forall i \in \mathcal{SW}, c \in \mathcal{C}, t \in \mathcal{T}, u \in \mathcal{U} \tag{3}$$

$$\sum_{j \in \mathcal{SW}} e_{i,j} \bar{w}^{t,u,c}_{i,j} - \sum_{j \in \mathcal{SW}} e_{j,i} \bar{w}^{t,u,c}_{j,i} = 0 \quad \forall i \in \mathcal{SW}, c \in \mathcal{C}, t \in \mathcal{T}, u \in \mathcal{U} \tag{4}$$

To ensure that each task is completed in a given deadline (based on D, θ, l, and π values for each task), the w rate from a proxy to the local OF switch, the w rate from the local OF switch to the Cloudlet, the $\bar{w}$ rate from the Cloudlet to the local OF switch and the $\bar{w}$ rate from the local OF switch to the BS should be provided the following constraints, respectively:

$$\theta_{t,u} \sum_{j \in \mathcal{SW}} e_{i,j} w^{c,t,u}_{i,j} = O^i_{t,u} l_t \rho_{u,i} \qquad \forall i \in \mathcal{P}, c \in \mathcal{C}, t \in \mathcal{T}, u \in \mathcal{U} \tag{5}$$

$$\theta_{t,u} \sum_{j \in \mathcal{SW}} e_{j,i} w^{i,t,u}_{j,i} = O^i_{t,u} l_t \qquad \forall i \in \mathcal{C}, t \in \mathcal{T}, u \in \mathcal{U} \tag{6}$$

$$\theta_{t,u} \sum_{j \in \mathcal{SW}} e_{i,j} \bar{w}^{i,t,u}_{i,j} = O^i_{t,u} l_t \pi_t \qquad \forall i \in \mathcal{C}, t \in \mathcal{T}, u \in \mathcal{U} \tag{7}$$

$$\theta_{t,u} \sum_{j \in \mathcal{SW}} e_{j,i} \bar{w}^{c,t,u}_{j,i} = O^i_{t,u} l_t \pi_t \rho_{u,i} \qquad \forall i \in \mathcal{A}, c \in \mathcal{C}, t \in \mathcal{T}, u \in \mathcal{U} \tag{8}$$

where l_t, π_t, and $\rho_{u,i}$ are defined as the data size of task t, the task resize coefficient after receiving services by the selected Cloudlet and the indicator representing the connection of user $u \in \mathcal{U}$ to BS $i \in \mathcal{A}$, respectively. In fact, constraints (5) to (8), due to the limited resources of the Cloudlets and the known limited deadline of each task, through the parameter $\theta_{t,u}$ forces the model to transfer the tasks data to the Cloudlets with the required data rate and vice versa.

The total traffic from each link $i, j \in \mathcal{V}$ for all the requests at time slot τ should not be greater than the available bandwidth in that link. Also, for the effective use of limited network resources, such as the bandwidth, the proposed model attempts to maximize the minimum normalized link utilization. Thus, we define variable R as the minimum bandwidth utilization wherein the objective function will be increased. For this purpose, the following constraint must be provided for each link:

$$b_{i,j} R \le b_{i,j} - \sum_{c \in \mathcal{C}} \sum_{u \in \mathcal{U}} \sum_{t \in \mathcal{T}} (w^{t,u,c}_{i,j} + \bar{w}^{t,u,c}_{i,j}) \qquad \forall i, j \in \mathcal{V} \tag{9}$$

where $b_{i,j}$ represents the available bandwidth between nodes i, $j \in \mathcal{V}$ at the beginning of each time slot τ. Note that the above equation considers the normalized bandwidth utilization by multiplying $b_{i,j}$ to R.

One of the main goals of the present research is to maintain a balanced and fair distribution of the load among all the Cloudlets, irrespective of the proximity of each user to the Cloudlet to which it is connected. To achieve this goal, this article defines variable M as the maximum rate of incoming traffic from the local switches to each Cloudlet. Therefore, minimizing M in the objective function results in load distribution among Cloudlets. In other words, this work achieves almost equal incoming traffic rates at all Cloudlets. To fulfill this requirement and to prevent unwanted traffic by the OF switches which wastes the resources, there should be the following constraint:

$$\sum_{t \in \mathcal{T}} \sum_{u \in \mathcal{U}} \sum_{j \in \mathcal{SW}} e_{i,j} (w^{i,t,u}_{j,i} \theta_{t,u}) \le M \mu_i \qquad \forall i \in \mathcal{C} \tag{10}$$

where μ_i is a total load of tasks assigned to the Cloudlet i at each given time slot. In Eq. (10), the value of variable M is normalized by multiplying μ_i.

Since the resources of each Cloudlet are limited and each task requires some of its resources to run, then the required resources for a task should be assigned when they are available. In fact, at each given time slot, the total required resources for the assigned tasks to a Cloudlet should not be greater than the available resources of that Cloudlet. If the execution of a task is not completed in a one-time slot, then the resources at the disposal of the task must be considered for the next time slots. For this purpose, the following constraint must be provided at each time slot τ.

$$\sum_{t \in \mathcal{T}} \sum_{u \in \mathcal{U}} O^i_{t,u} \xi^i_t \le \chi_i \qquad \forall i \in \mathcal{C} \tag{11}$$

In the constraint, $\xi^i_{t,u}$ and χ_i are the percentage of the resources of Cloudlet i assigned to task t of user u and the percentage of the available resources in each Cloudlet $i \in \mathcal{C}$ at each time slot, respectively.

According to the objectives described in the problem description (section.3), the proposed model is defined as a multi-objective function for fair load distribution among the Cloudlets, the optimization of network capacity utilization and maximization of the throughput for the mobile user requests as follows:

$$\text{Minimize} \quad \alpha M - \beta R - \gamma \sum_{c \in \mathcal{C}} \sum_{u \in \mathcal{U}} \sum_{t \in \mathcal{T}} \frac{O^c_{t,u}}{\Upsilon} \tag{12}$$

$$\text{s.t.} \qquad \text{Constraints Eqs. (1)–(11)}$$

$$\text{vars.} \qquad O^c_{t,u} \in \{0, 1\}, w^{t,u,c}_{i,j}, \bar{w}^{t,u,c}_{i,j}, M, R \ge 0$$

where $\alpha + \beta + \gamma = 1$. In the proposed MILP model, the $O^c_{t,u}$ variable is normalized by dividing the total number of the requested tasks in each time slot, which is shown by Υ. At each time slot τ, model (12) runs for the balanced and fair load distribution and optimal determination of the data transmission rate of the tasks.

The current work can also determine appropriate priorities for criteria in the objective function by selecting weight coefficients α, β, and γ. For example, β helps to determine the priority of bandwidth consumption in network links. Various weights indicate the preference of three objectives while optimizing the problem. These weights coefficients can be set based on the provider's policies at each time slot τ.

Although the objective function is linear, and the binary variables $O^c_{t,u}$, render the model a MILP which is generally NP-hard and unsolvable in polynomial time.

Theorem 1. *The proposed model (12) is an NP-hard problem and is challenged by high time complexity.*

Proof. The objective is to decide whether the tasks are assigned to the Cloudlets for the purpose of maximizing the system throughput in the MILP problem. Without loss of generality, supposing that a set of identical Cloudlets indexed by $\mathcal{C}$ = {1, 2…i} and a set of tasks denoted by $\mathcal{Y}$ = {1, 2… j} which should be assigned to Cloudlets in a time slot t. Moreover, since each task j requires a specific amount of resources, we can consider weight w_j for each $j \in \mathcal{Y}$. Furthermore, it is assumed that each Cloudlet i has a k-type resource (e.g., CPU, memory, bandwidth) with the maximum i_k capacity. Considering each resource k in a Cloudlet could be treated as one dimension of a k-dimension bin with a capacity i_k, the addressed problem can be reduced to the well-known bin packing problem in polynomial time. It is worth noting that the objective function purpose to maximize its total gain by increasing service rate (i.e., system throughput) and reducing resource usage. Since the bin packing problem is NP-hard, it concludes that the addressed problem is not solvable in a polynomial time and is in the form of NP-hard [27]. ■

Due to the high time complexity of the proposed MILP model (12), it would be impractical to employ the centralized approach in a real scenario. To mitigate this drawback, we propose certain modifications, which remove the binary variables and allow the problem to take the form of an LP so that the SDN controller can centrally run it on the large-scale networks. For this purpose and without loss of the problem's generality, it is assumed that all the Cloudlets are homogeneous in delivering the services, but heterogeneous where the amount of the resources is concerned.

Additionally, the user tasks are divided into K types according to the required service ($k = 1, 2, 3\ldots$, K). The tasks of the same type have the same basic characteristics and parameters in terms of size, deadline, etc. Each user can be sent r tasks of type $k \in K$ to the network at each time slot τ. We first define the variable $x^k_{i,j}$ as the number of k-type tasks from all users connected to the BS i assigned to the Cloudlet j. Therefore, instead of using binary variable $O^c_{t,u}$ it can select the number of k-type tasks to be assigned to each Cloudlet. Now, by performing some minor modifications on the proposed MILP model, the LP relaxation model can be represented as follows:

$$\textit{Minimize}\quad \alpha M - \beta R - \gamma \sum_{j\in\mathcal{C}}\sum_{i\in\mathcal{A}}\sum_{k\in\mathcal{K}} \frac{x^k_{i,j}\phi_k}{\Upsilon} \tag{I}$$

where Υ is the total number of the requested tasks in each time slot τ as well as $\alpha + \beta + \gamma = 1$. In the LP relaxation model (I), the weight coefficient ϕ_k determines the priority of accepting the tasks of a particular type. This coefficient can be adjusted according to the operator's policies in each time slot τ. **s.t.**

$$\sum_{j\in\mathcal{SW}} e_{i,j} w^{c,k}_{i,j} - \sum_{j\in\mathcal{SW}} e_{j,i} w^{c,k}_{j,i} = 0 \qquad \forall i \in \mathcal{SW}, c \in \mathcal{C}, k \in \mathcal{K} \tag{II}$$

$$\sum_{j\in\mathcal{SW}} e_{i,j} \bar{w}^{c}_{i,j} - \sum_{j\in\mathcal{SW}} e_{j,i} \bar{w}^{t,u,c}_{j,i} = 0 \qquad \forall i \in \mathcal{SW}, c \in \mathcal{C}, k \in \mathcal{K} \tag{III}$$

$$\theta_k \sum_{j\in\mathcal{SW}} e_{i,j} w^{c,k}_{i,j} = x^k_{i,j} l_k \qquad \forall i \in \mathcal{P}, k \in \mathcal{K}, c \in \mathcal{C} \tag{IV}$$

$$\theta_k \sum_{j\in\mathcal{SW}} e_{j,i} w^{i,k}_{j,i} = \sum_{j\in\mathcal{A}} x^k_{j,i} l_k \qquad \forall i \in \mathcal{C}, k \in \mathcal{K} \tag{V}$$

$$\theta_k \sum_{j\in\mathcal{SW}} e_{i,j} \bar{w}^{i,k}_{i,j} = \sum_{j\in\mathcal{A}} x^k_{j,i} l_k \pi_k \qquad \forall i \in \mathcal{C}, k \in \mathcal{K} \tag{VI}$$

$$\theta_k \sum_{j\in\mathcal{SW}} e_{j,i} \bar{w}^{c,k}_{j,i} = x^k_{j,i} l_k \pi_k \qquad \forall i \in \mathcal{A}, k \in \mathcal{K} \tag{VII}$$

$$b_{i,j} R \le b_{i,j} - \sum_{c\in\mathcal{C}}\sum_{k\in\mathcal{K}} (w^{c,k}_{i,j} + \bar{w}^{c,k}_{i,j}) \qquad \forall i, j \in \mathcal{V} \tag{VIII}$$

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{SW}} e_{i,j}(w^{i,k}_{j,i}\theta_k) \le M\mu_i \qquad \forall i \in \mathcal{C} \tag{IX}$$

$$\sum_{j\in\mathcal{C}} x^k_{i,j} \le \psi^k_i \qquad \forall i \in \mathcal{A}, k \in \mathcal{K} \tag{X}$$

$$\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{A}} x^k_{i,j}\xi^j_k \le \chi_i \qquad \forall j \in \mathcal{C} \tag{XI}$$

$$\textit{vars.} \qquad x^k_{i,j},\ w^{c,k}_{i,j},\ \bar{w}^{c,k}_{i,j},\ M, R \ge 0$$

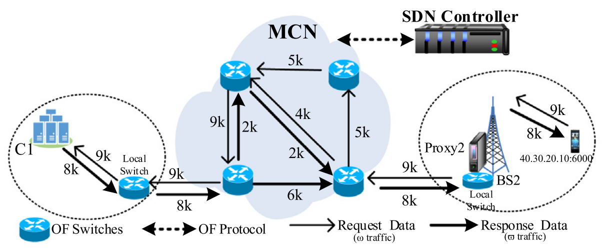


Fig. 7. An example of data traffic-offloading task (w) and its response ($\bar{w}$).

where l_k and π_k in the constraints (V) to (VI) are defined as the data size of the k-type task and the task resize coefficient after servicing by the selected Cloudlet, respectively. Also, μ_i in the constraint (IX) and ψ^k_i in the constraint (X) are a total load of tasks assigned to the Cloudlet i and the number of k-type tasks for all the users connected to BS i in a time slot τ, respectively. Further, ξ^i_k and χ_i in the constraint (XI) are the percentage of the resources of Cloudlet i assigned to a k-type task and the percentage of the available resources in each Cloudlet $i \in \mathcal{C}$ at each time slot, respectively.

Constraint (X) ensures that the number of k-type tasks for a BS i, which is assigned to all the Cloudlets is not greater than the number of the k-type tasks received in that BS.

Flow Programmer Module (FPM): This module performs the final step to launch data flows from local OF switches to Cloudlets and vice versa through MCN by executing the necessary commands from the SDN controller to the OF switches by OF protocol [9]. The FPM module provides optimal data transfer rates $w^{c,k}_{i,j}$ and $\bar{w}^{c,k}_{i,j}$ for each type of task k for node i to $j \in \mathcal{V}$ from the FOM module.

As SDN switches support queues so as to provide desirable data rate calibration on each flow [28], configuring different queues on all OF switches may seem like an overly simple approach and rather naive as there are numerous tasks with different data rates that must be delivered in each time slot. Moreover, by increasing the number of switches, the probability of misconfiguration grows. Therefore, setting up data paths in each time slot requires an agile and reliable method with low complexity. Fig. 7 presents an example of how the data transfer between UE and Cloudlet may be achieved. In this figure, via socket 40.30.20.10:6000, proxy2 receives a request from a user connected to it. Then, the proxy suspends the connection to the relevant user and sends a request to the SDN controller. After processing the request by the TAM module and extracting the key parameters of the request, the FOM module specifies that the request should be executed by c1. Hence, the data packets of this request must be sent from proxy2 to c1.

The FOM module specifies the routes and the optimal data rates required to send the task and then receive the response. Now, with the optimal rates, i.e., $w^{c,k}_{i,j}$ and $\bar{w}^{c,k}_{i,j}$ for k-type task, the SDN controller sends the required rules for the OF switches. Fig. 7 indicates the output of the FOM module based on the available capacity of each link in the current time slot for the requested task-traffic rate for each link. In [29], we have provided an efficient way to configure the OF switches and route data flows from the proxy to the Cloudlets and from the Cloudlet to the BS based on the $w^{c,k}_{i,j}$ and $\bar{w}^{c,k}_{i,j}$ rates obtained from the FOM module output.

5. Performance evaluation of the proposed models

In order to validate our proposed approach, the work has implemented the experiments using Python 2.7 and Pulp library

Table 2
The characteristics of incoming tasks to the system.

Type of task	Parameters of task					
	Number of slots needed per task (n)	Task size (l) (Mb)	Deadline for task completion (D) (per second)	Priority of task execution (ϕ)	Required resources per time slot (ξ)	Task resizing after execution (π)
$k1$	2	8	0.9	0.5	0.6	0.8
$k2$	1	6	0.6	0.5	0.9	0.5
$k3$	3	5	0.5	0.5	0.5	0.4
$k4$	5	7	0.7	0.5	0.2	0.6

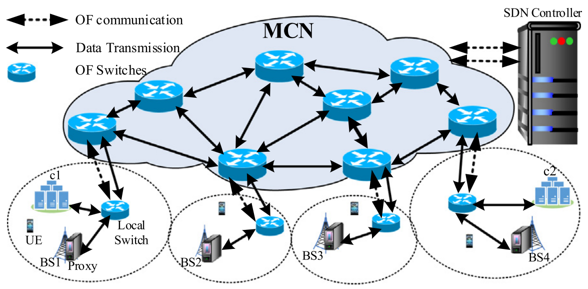


Fig. 8. The network topology to evaluate the proposed model.

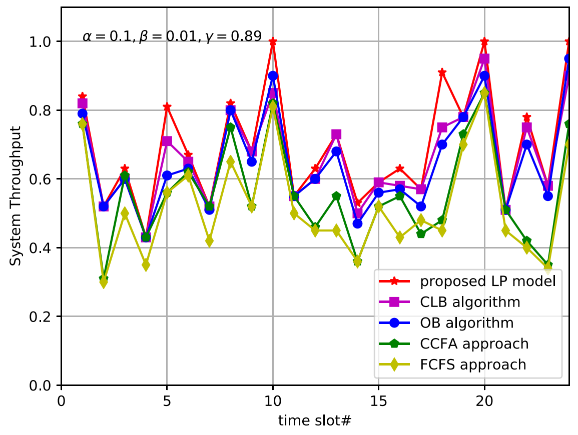


Fig. 9. The system throughput of algorithms CLBA, CCFA, FCFS, OB, and the proposed LP model.

1.6 tools [30]. The article has utilized Mininet-WiFi [31] to generate the network topology and perform the evaluations. The researchers also employed Floodlight as the SDN controller [32] in these experiments. The experiments in this study fundamentally consisted of two parts: (1) a comparison of the proposed LP model with other well-known approaches; (2) an investigation of the proposed LP model parameters.

5.1. Basic setups

The study is based on a multi-Cloudlet environment that consists of 2 Cloudlets, four BSs, eight OF switches and a set of mobile devices sending their requests to the Cloudlets for processing as depicted in Fig. 8. The radio coverage area for each BS is 1 km × 1 km. In addition, the bandwidth available between each of these components is taken into account according to the defined scenario. Moreover, it is assumed that all the Cloudlets are capable of serving four types of tasks $k1$, $k2$, $k3$, and $k4$ to serve users and are homogeneous in terms of the available resources. Any user within the range of any BS can request any of the above services (any user can have one request per service in any time slot). The number of input tasks for each type from each BS is selected randomly ranging from 0 to 100 (via a random generator in Python), which means that up to a maximum of 100 users per time slot can be in the range of each BS. The details of the four types of services are presented in Table 2 . These features include the number of slots required to complete a task (n), the size of each task (l), the deadline time (D), the priority of running a service (ϕ), the resources required per task (ξ) (as a percentage of each Cloudlet's resources), and the resizing coefficient of the service after it is executed (π). The characteristics of each service type for all incoming tasks are assumed to be the same at all times during the experiments.

The proposed LP model is run in an Lenovo laptop with Intel Core i5 processor and 4 GB RAM. In addition, the work has repeated the simulations 100 times and measured the average results. In each scenario, the simulation is run for 25 time slot.

5.2. Evaluation criteria

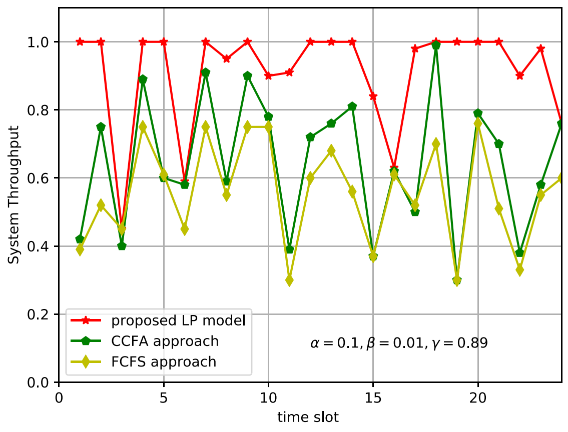
In order to evaluate the proposed LP model and compare them with the basic algorithms and existing research, the following criteria are introduced:

1. Throughput (i.e., the overall system service rate): this criterion refers to the ratio of accepted requests in all the Cloudlets to the total number of requests to the system per time slot. The higher system throughput means higher operator revenue and higher user satisfaction.
2. Input load balancing to the Cloudlets: a balanced distribution of requests will prevent some Cloudlets from being overloaded, which in turn will increase the execution time of the requests and ultimately reduce the QoS.
3. Optimal use of the communication link capacity: optimal use of the network resources will prevent congestion at the mobile core network and ultimately improve the QoS.
4. Scalability: executing an algorithm or model in a short time with a large number of Cloudlets or a mass number of users makes it possible to implement in a real network.

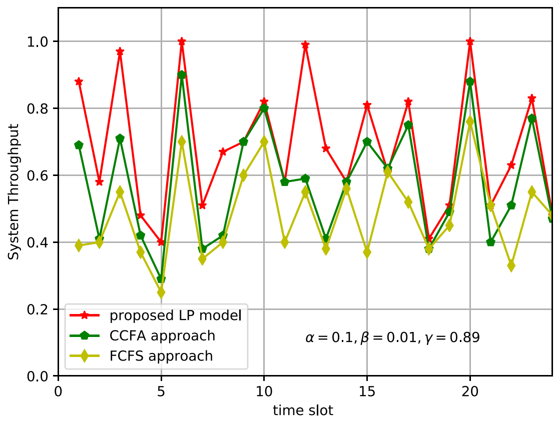
5.3. Comparison of the proposed LP model against the benchmark algorithms

In the present study, the proposed LP model has been evaluated against the benchmark algorithms such as First-Come-First-Service batch (FCFS), CCFA-Batch, Online-Batch (OB) [7], and Cloudlet Load Balancing algorithm (CLBA) in a wireless metropolitan area network (WMAN) [11] in terms of the system throughput. For the OB algorithm, we considered the values of parameters B and B_k at the same as their value in paper [7]. For the CLBA, CCFA, and OB, for each user, the closest Cloudlet to any user is considered as a home Cloudlet.

For evaluation, this work has implemented the topology depicted in Fig. 8 on the Mininet-wifi environment. Moreover, the link bandwidth between system elements is considered high (1 Gb) to avoid limiting the acceptance of tasks. Then, by sending requests to the proposed LP model and other algorithms with the specifications of Table 2, their throughput has been calculated.



(a)



(b)

Fig. 10. The effect of increase in the resources of a Cloudlet (a), and increase in the required resources for each task (b) on the System throughput.

Table 3
The number of needed time slots for each task in different intervals.

Number of time sltos	Intervals							
	1	2	3	4	5	6	7	8
For $k1$-type task	1	1	2	2	3	4	5	5
For $k2$-type task	1	2	3	4	5	6	4	6
For $k3$-type task	2	3	2	1	2	2	1	3
For $k4$-type task	3	5	1	4	4	3	5	6

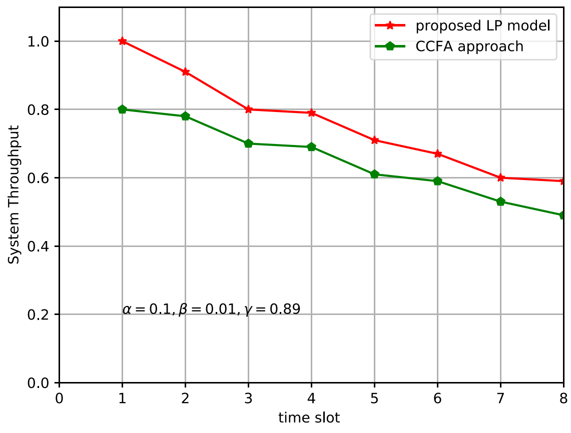


Fig. 11. The effect of the number of required time slots for each type of task on the system throughput.

The performance curves of these algorithms are plotted in Fig. 9. It is shown that the proposed LP model outperforms the counterparts CLBA, FCFS-Batch, CCFA-Batch, and Online-Batch in terms of the system throughput over different time slots.

For example, the system throughput of the proposed LP model is around 5% higher than that of the CLBA, 10% higher than that of the OB, 17% higher than that of the CCFA algorithm, and 21% higher than that of the FCFS algorithm. The rationale behind this is that the proposed model attempts to accept the most possible requests in all Cloudlets at each time slot. Consequently, the requests that require a lot of resources are automatically rejected if resources are not available in the Cloudlets. Whereas, in the OB Algorithm, at the beginning of the entry of requests, the tasks that a large quantity of resource demand beyond the threshold level are to be rejected even if the home Cloudlet of that request is under-loaded. In contrast, the other two algorithms (the CCFA and FCFS) have no mechanism for rejecting the tasks requiring high resources, and this, coupled with the admission of tasks only in the nearest Cloudlet, has reduced the overall system throughput.

In the CLBA algorithm, each Cloudlet accepts all input tasks from its home users until the request deadline is met. When requests grow and the Cloudlet does not have the sufficient available resources to fulfill the requests in a given time period, the requests are sent to the under-loaded Cloudlets. Meanwhile, selecting the best Cloudlet for submission is based on the minimum response time of the Cloudlets which is accomplished by the distributed algorithm at each node. Accordingly, the high convergence time and high traffic load will affect the system throughput. The lack of a mechanism for rejecting tasks requiring a lot of resources, especially when the Cloudlet is in high load will result in reducing the system throughput compared to the proposed LP model.

5.3.1. *The effect of changing parameters on the system throughput*

The next experiment evaluates the impact of the task parameters changes on the system throughput. Accordingly, in the first step, the second Cloudlet resources are doubled and its impact on the system throughput has been examined. Fig. 10(a) indicates that the proposed model significantly outperforms the CCFA and FCFS algorithms in terms of the overall system throughput. The rationale for this increase for the proposed LP model is that the pool of shared resources for distributing user tasks has become larger. While for the other two approaches, the resources of one Cloudlet has not increase.

Moreover, in order to observe the effect of increasing the required resources of each task on the system throughput, this article has increased the parameter ξ from 0.6 to 1.2 (for $k1$) and from 0.9 to 1.8 (for $k2$). Fig. 10(b) indicates that altering these parameters directly impacts the number of consumed resources in each Cloudlet. As expected, the overall system throughput falls by increasing the needed resources for each task in each Cloudlet. Nevertheless, the system throughput in the proposed model is much greater than that of the CCFA and FCFS algorithms.

In consideration of the high bandwidth links, the overall system throughput is not affected by changing the parameters of the task size (l) nor by the deadline of each task (D) in the selected Cloudlet. The reason is that the constraints (V) to (VI) attempt to increase the traffic transfer rate by strain on the model, in order to increase the usage of the link capacity. Consequently, the transfer delay and execution of a task are reduced and the deadline for the task is fulfilled.

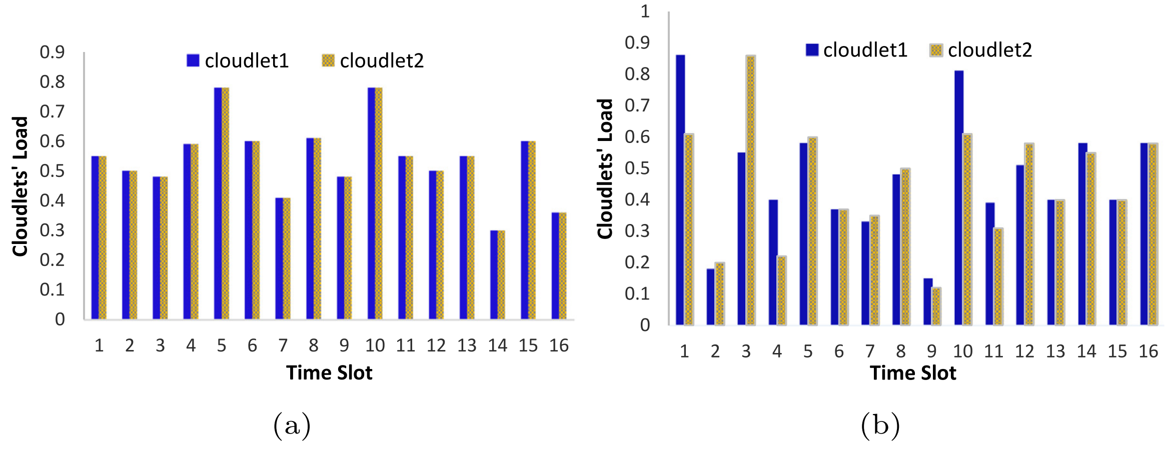


Fig. 12. The balanced load distribution among the Cloudlets (a) the high bandwidth links and (b) low bandwidth links ($\alpha = 0.49$, $\beta = 0.01$, $\gamma = 0.5$).

As mentioned earlier, the execution of a task may not be completed within a time slot of a dedicated Cloudlet and the execution also consumes a part of Cloudlet resources in subsequent slots. This occurs for computation-intensive tasks and whose Cloudlet resources are taken and are not freed for multiple time slots. This factor can directly affect the acceptance of incoming tasks in subsequent time slots and can also increase the number of rejected tasks. To investigate this, the present study implements the network topology depicted in Fig. 8 with the parameters in Table 3 along with the number of required different time slots for each type of task given in Table 2. In addition, each interval contains 24 time slots.

The output curve is plotted in Fig. 11. As can be observed from the figure, when the number of time slots required for tasks increases, the average system throughput declines. This is due to a long occupation of Cloudlet resources which increases the number of rejected tasks. It is grave to note that, although the average system throughput lowers by the increasing number of needed time slots, the average throughput of the proposed LP model is higher compared to the other algorithms. The reason for this, as in previous experiments, is the better distribution of tasks among the shared resources.

5.4. The load balancing among the cloudlets

Load balancing signifies that all Cloudlets are equal in terms of the number of resources used to allocate tasks. This assists to increase the system overall throughput because each Cloudlet can handle most of the tasks related to users nearby and has no need to send any task to other Cloudlets through network links. In fact, a balanced load distribution facilitates the optimal utilization of network link bandwidth. As the criterion for the distribution of load balance, this section considers that the number of tasks accepted in each Cloudlet is multiplied by the number of resources consumed by each tasks in each time slot τ. Considering the values of Table 2 and assuming the values of α, β, and γ as 0.49, 0.01, and 0.5, respectively, the system output as in Fig. 12 illustrates that the load rate assigned to Cloudlets has been fairly met both with (a) the high bandwidth links (b) and low bandwidth links, while using the maximum available Cloudlet resources.

In the second scenario, the optimal use of the Cloudlets shared resources is not possible due to the limited capacity of the links. Nevertheless, the average load difference of Cloudlets in this scenario is less than 8% and indicates that the variable M is correctly defined in the model.

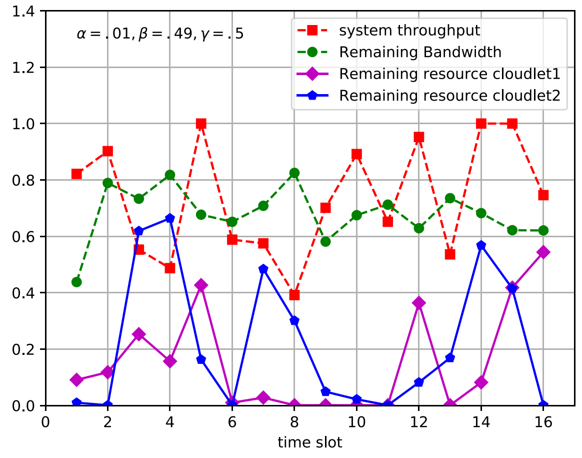


Fig. 13. The system throughput, consumed bandwidth of links, and utilized resources in the Cloudlets.

5.5. Optimal use of link bandwidth

In the current work, the purpose of optimal use of link bandwidth is to consume less bandwidth when assigning tasks to the Cloudlets which also have the highest system throughput. This signifies that, while the present study is able to allocate the most tasks to Cloudlets with limited resources and maximize the system throughput, it can also maximize the bandwidth utilization of network links in many limited cases. For this purpose, the proposed LP model is simulated with the topology depicted in Fig. 8 and the parameters introduced in Table 2, with links of 100Mb between the OF switches and 1 Gb between the BSs, Cloudlets, and local OF switches. The overall system throughput is determined by the consumed bandwidth and the quantity of consumed resources per Cloudlet at each given time slot. Fig. 13 has depicted the output results for this scenario.

As expected, the model attempts to minimize the bandwidth consumption of links as much as possible through the achievement of maximum throughput and usage of all Cloudlet resources. As observed in Fig. 13, while the overall system throughput is 1 for time slots 5, 14, and 15, the Cloudlet resources are not completely utilized and the link bandwidth consumption is less than 30%. In general, the average system throughput is more than 70%; thus, it is indicating the efficiency of the proposed LP model.

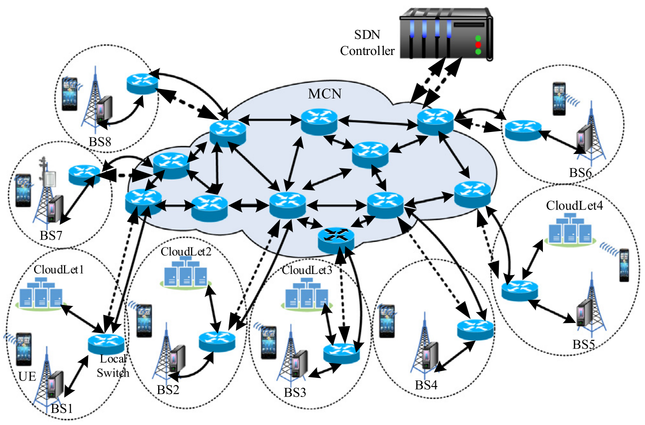


Fig. 14. The network topology for medium-scale network.

5.6. Scalability evaluation of the proposed LP model

The scalability of a model is one of the most critical parameters to implement on a large scale network. To this end, the work has added four BSs and two Cloudlets to the network topology in Fig. 8 and has examined their impact on the system performance (see Fig. 14).

This subsection presents three scenarios for evaluating the expansion of the scope of the Cloudlet network into the system throughput and investigates them in detail. To do so, the two scenarios A and B, have evaluated for a medium-scale network in terms of the number of BSs, Cloudlets, and users. Then, in scenario C, by increasing the number of users and Cloudlets in the system, we calculated the time complexity of the MILP and LP models in terms of *running time* and showed that the proposed LP model is suitable for the large-scale networks.

Scenario A: It is initially assumed that only two Cloudlets are added to the network topology and that the added BS does not send any requests to the network. In fact, it is intended to demonstrate that, as the number of Cloudlets increases and with the same number of requests as in the previous sections, the overall system throughput improves. This will result in improved user satisfaction through the acceptance of more tasks and a reduction in the number of rejected tasks. For this purpose, the model is implemented with the parameters in Table 2 similarly to the procedure in Section 5.3. Consequently, as the number of Cloudlets rises, the resources of the entire network grow and the system throughput sharply increases, with an average of about 0.95 (as depicted in Fig. 15(a)). In this case, compared to Section 5.3, the system throughput for the CCFA and FCFS strategies are unchanged, because, although the number of Cloudlets has increased, the requests from each BS are only sent to the nearest Cloudlet and no other Cloudlet resources are applies.

Scenario B: The current work considers the network topology of Fig. 14 along with the BSs and the Cloudlets. It also supposes that there are incoming tasks for each type of task from each BS between 1 to 100 (assuming that, at each time slot τ, a maximum of 3200 tasks enter the system). The values of all parameters are considered to be the same as those of Table 2. Fig. 15(b) has depicted the output of the results. As expected, the system throughput has improved compared to Section 5.3. This increase in performance is due to the fact that the pool of shared resources is larger for distributing tasks and the use of Cloudlets with free resources is provided. Despite the increase of BSs, the presence of Cloudlets in the network environment results in more efficient distribution of input loads. This increases the system throughput, reduces the number of rejected tasks and ultimately increases user satisfaction.

Scenario C: To analyze the time complexity of the two proposed models, by increasing the number of BSs and Cloudlets to 500 and 50, respectively, the paper has implemented two models with the number of incoming requests ranging from 10 to 20000 using the pulp library in python. The LP model is run at all-time slots ranging from 250 to 450 ms. Yet the MILP model with 10 to 70 tasks has illustrated exponential growth of 320 to 950 ms. Consequently, these results have proved that the LP model actually works well on large-scale networks (see Fig. 15(c)).

5.7. The comparison of the MILP with the LP model

The MILP model has a high time complexity, thus, it is not suitable for implementation in large-scale networks. Therefore, the current study proposes a LP-relaxed model that provides a solution in a short time. It is necessary, however, to show that the output of the LP model is proportional to the original MILP model. For this purpose, based on Fig. 14 and the parameters of Table 2, this work has evaluated two models with the same inputs.

The output curve of the system throughput for the both models is depicted in Fig. 16(a). As can be observed, the behavior of the proposed LP model is equivalent to the original MILP model in most of the time slots.

Another issue for which these two models should be compared is the objective function. For this purpose, the topology of Fig. 14 is again implemented along with the parameters of Table 2 considering the same incoming tasks. In Fig. 16(b), it can be observed that the LP model follows the MILP model, satisfactorily.

5.8. The role of objective function coefficients in the system performance

According to issues described in the fourth section, the proposed LP and MILP models are multi-objective functions. In these models, the coefficient of each part effects on overall system performance based on its significant. Therefore, the value of the coefficients must be determined and performed for each time slot based on the operator's policies. In fact, the operator regulates the system performance using the coefficients according to the user requirements, its own requirements, and network constraints. In order to evaluate the influence of these coefficients on the overall system performance, the proposed LP model is simulated with the topology depicted in Fig. 8 and the parameters introduced in Table 2. Experiments have conducted with high and low capacity links scenarios. In addition, for any value of coefficients, the simulation is run for 25 time slot. The work has repeated the simulations 100 times and measured the average results.

The coefficient α indicates the importance of the input load balance of Cloudlets (as the variable M) in the objective function. Fig. 17(a) shows the results of this experiment (α: 0.1–0.8) and the α effect on load balancing with value $\beta = 0.1$. This figure confirms by increasing the α value, the system moves toward a better load balance (less input load difference between Cloudlets). In the first scenario, it is possible to accept the requests in each Cloudlets due to the high bandwidth of the links. Hence, the input load difference between Cloudlets is zero from $\alpha = 0.6$ onwards.

In the second scenario, there is a low bandwidth of the links and the inability to transmit task data between the Cloudlets. Therefore, by increasing the α value, their input load difference is decreased with the rejection of the requests which negatively affects the system throughput (see Fig. 17(b)). Also, in the first scenario, by increasing the α value (i.e., decreasing the γ value), the overall system throughput increases until $\alpha = 0.4$. In this

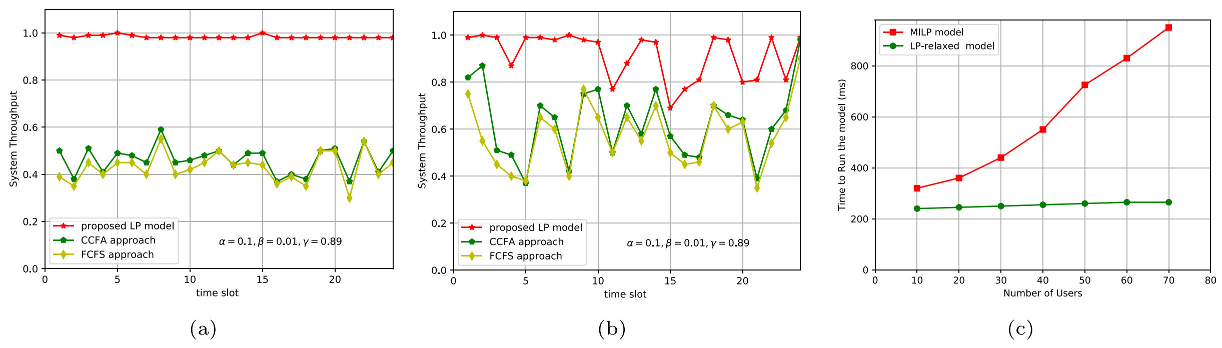


Fig. 15. The results of scenario A, B, and C for scalability and time complexity.

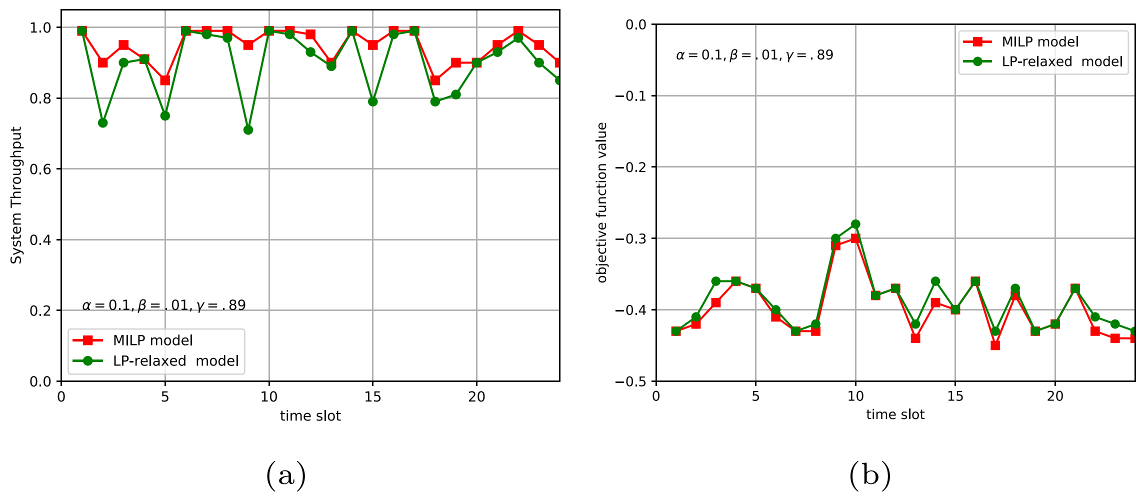


Fig. 16. Comparison of the MILP and LP models for (a) the overall system throughput and (b) the objective function value.

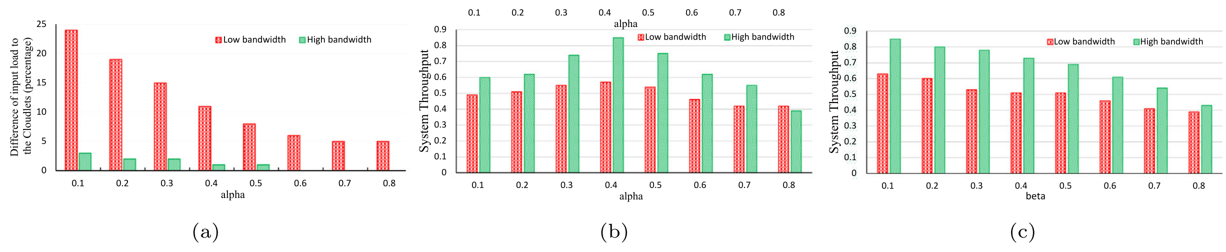


Fig. 17. The role of objective function coefficients in the system performance (a) effect of increasing α on the input load difference between Cloudlets (b) effect of increasing α on the system throughput (c) effect of increasing β on the links capacity utilization.

point (i.e., $\gamma = 0.5$) the model attempts to maximize the overall system throughput by the load balancing at the entrance of Cloudlets.

In the values of $\alpha \geq 0.5$ (i.e., $\gamma \leq 0.4$), the system throughput and the input load difference of the cloudlets have simultaneously decreased. The reason for this is that by reducing the value of α, the load balancing is more important than the system throughput. Therefore, to make a trade-off between the load balance and the system throughput, $\alpha = 0.4$ and $\gamma = 0.5$ are the most appropriate values.

The coefficient β has been considered in the objective function to prioritize the links capacity utilization between system components (as a variable R). By increasing this factor, the operator can reduce the use of communication links and prevent congestion in the MCN. But this policy can negatively affect the overall system throughput (see Fig. 17(c)) (with value $\alpha = 0.1$). However, this effect is more numerous in the first scenario. Whereas, in the second scenario, the capacity limitation of the links reduces the system throughput, and increasing β has not much effect on system behavior. In the first scenario, with the increase of β, the system's tendency decreases to use the capacity of the links, which negatively affects the acceptance of the tasks. So that at the $\beta = 0.8$, the system throughput is nearing the second scenario. The results of the experiments show that the network operator must select the above coefficients with high accuracy and in accordance with the network conditions and user demands. For instance, to maximize load balancing and prevent overloading in some Cloudlets, which may reduce the quality of applications running in them, the α value should be large. However, a large value for α means fewer values for β and γ, which means lower system throughput and the greater usage of network link capacity. In another case, by increasing the β value, network link capacity becomes more important. Therefore, the model intends to reduce congestion in the MCN by using fewer capacity of links. However, this increase will prevent task acceptance and ultimately reduce the system throughput.

6. Conclusions and future work

Proper Cloudlet selection in a multi-Cloudlet environment and optimal resource management in allocating requests to the Cloudlets is a promising research area that has commanded significant attention from researchers.

In this paper, we have proposed a framework for a task deadline-awareness balanced distribution of tasks across the Cloudlets by leveraging SDN technology with the aim of maximizing the system throughput. Then, the problem is formulated as a MILP model, which features optimal data paths for delivering user-requested tasks to the Cloudlets. we have presented that it is NP-hard and is not suitable for large scale environments due to its high time complexity. The MILP model has been relaxed to an LP model to enable large-scale networking by exploiting a centralized approach and overview of system resources. Then, the proposed LP model has been implemented with the help of Python and the Pulp Library in the Mininet-wifi environment using the Floodlight controller.

Experimental results illustrate that the proposed LP model increases the overall system throughput by approximately 5%–21% compared to the OB and CLBA algorithms and the CCFA and FCFS baseline approach.

Hence, the researches can recommend the proposed model for the problem of balanced load distribution with the deadline-awareness of each task in the SDN based Cloudlet network in order to manage the resources optimally and maximize the system throughput.

As a future research direction, we wish to extend the proposed work in scenario with the mobility of a user and the problem of minimizing the costs for running applications for each user along the entire route.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper, Cisco.
[2] B. Varghese, R. Buyya, Next generation cloud computing: New trends and research directions, Future Gener. Comput. Syst. 79 (2018) 849–861.
[3] E. Ahmed, M.H. Rehmani, Mobile Edge Computing: Opportunities, Solutions, and Challenges, ed: Elsevier, 2017.
[4] J. Du, C. Jiang, H. Zhang, Y. Ren, M. Guizani, Auction design and analysis for SDN-based traffic offloading in hybrid satellite-terrestrial networks, IEEE J. Sel. Areas Commun. 36 (10) (2018) 2202–2217.
[5] ETSI Mobile Edge Computing Portal. http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing.
[6] R. Olaniyan, O. Fadahunsi, M. Maheswaran, Opportunistic edge computing: concepts, opportunities and research challenges, Future Gener. Comput. Syst. 89 (2018) 633–645.
[7] M. Saad, Fog computing and its role in the internet of things: Concept, security and privacy issues, Int. J. Comput. Appl. 180 (2018) 7–9.
[8] Q. Xia, W. Liang, W. Xu, Throughput maximization for online request admissions in mobile Cloudlets, in: Local Computer Networks (LCN), 2013 IEEE 38th Conference on, 2013, pp. 589–596.
[9] Q. Xia, W. Liang, Z. Xu, B. Zhou, Online algorithms for location-aware task offloading in two-tiered mobile cloud environments, in: Utility and Cloud Computing, 2014 IEEE/ACM 7th International Conference on, pp. 109–116.
[10] OpenFlow Switch Specification v1.0-v1.4; www.opennetworking.org/sdn-resources/onf-specifications.
[11] K. Benzekki, A. El Fergougui, A. Elbelrhiti Elalaoui, Softwaredefined networking (SDN): a survey, Secur. Commun. Netw. 9 (2017) 5803–5833.
[12] M. Jia, J. Cao, W. Liang, Optimal Cloudlet placement and user to Cloudlet allocation in wireless metropolitan area networks, IEEE Trans. Cloud Comput. 5 (2017) 725–737.
[13] M.Z. Nayyer, I. Raza, S.A. Hussain, A survey of Cloudlet-based mobile augmentation approaches for resource optimization, ACM Comput. Surv. 51 (5) (2018) 1–28.
[14] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for vm-based Cloudlets in mobile computing, IEEE Pervasive Comput. 8 (2009).
[15] V. Cardellini, V.D.N. Personé, V. Di Valerio, F. Facchinei, V. Grassi, F.L. Presti, et al., A game-theoretic approach to computation offloading in mobile cloud computing, Math. Program. 157 (2016) 421–449.
[16] M. Jia, W. Liang, Z. Xu, M. Huang, Cloudlet load balancing in wireless metropolitan area networks, in: INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE, 2016, pp. 1–9.
[17] L. Liu, Q. Fan, Resource allocation optimization based on mixed integer linear programming in the multi-Cloudlet environment, IEEE Access 6 (2018) 24533–24542.
[18] L. Zhao, W. Sun, Y. Shi, J. Liu, Optimal placement of Cloudlets for access delay minimization in SDN-based Internet of Things networks, IEEE Internet Things J. 5 (2018) 1334–1344.
[19] Z. Xu, W. Liang, W. Xu, M. Jia, S. Guo, Efficient algorithms for capacitated Cloudlet placements, IEEE Trans. Parallel Distrib. Syst. 27 (2016) 2866–2880.
[20] X. Sun, N. Ansari, PRIMAL: Profit maximization avatar placement for mobile edge computing, in: Communications (ICC), 2016 IEEE International Conference on, 2016, pp. 1–6.
[21] Q. Fan, N. Ansari, X. Sun, Energy driven avatar migration in green Cloudlet networks, IEEE Commun. Lett. 21 (7) (2017) 1601–1604.
[22] L. Tong, Y. Li, W. Gao, A hierarchical edge cloud architecture for mobile computing, in: 35th Annual IEEE Intl. Conf. on Comp. Comm. (INFOCOM 2016), San Francisco, CA, 2016, pp. 1–9.
[23] A. Gasmelseed, R. Ramalakshmi, Performance analysis of centralized and distributed SDN controllers for load balancing application, in: 2nd International Conference on Trends in Electronics and Informatics (ICOEI), IEEE, 2018.
[24] E. Daraghmi, Sh. Yuan, A small world based overlay network for improving dynamic load- balancing, J. Syst. Softw. 107 (2015) (2015) 187–203, http://dx.doi.org/10.1016/j.jss.2015.06.001.
[25] M. Karakus, A. Durresi, A survey: Control plane scalability issues and approaches in software-defined networking (sdn), Comput. Netw. 112 (2017) 279–293.
[26] J. Liu, Y. Li, D. Jin, SDN-based live VM migration across datacenters, ACM SIGCOMM Comput. Commun. Rev. 44 (4) (2014) 583–584.
[27] R.G. Michael, S.J. David, Computers and Intractability: A Guide to the Theory of NP-Completeness, WH Free. Co., San Fr, 1979, pp. 90–91.
[28] Open vSwitch, Available: http://docs.openvswitch.org/.
[29] F. Tashtarian, A. Erfanian, A. Varasteh, S2VC: An SDN-based framework for maximizing QoE in SVC-based HTTP adaptive streaming, Comput. Netw. 146 (2018) 33–46.
[30] Optimization with PuLP, Available: https://pythonhosted.org/PuLP/.
[31] B. Lantz, B. Heller, N. McKeown, A network in a laptop: rapid prototyping for software-defined networks, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, 2010, p. 19.
[32] Floodlight Controller, Available: http://www.projectoodlight.org/.

Shirzad Shahryari received the B.Sc. degree in electronic engineering from the Malek Ashtar University of Technology, Shahin shahr Isfehan, Iran, in 1994, and the M.Sc. degree in information and communication technology (ICT) from the Iran University of Science and Technology (IUST), Tehran, Iran, in 008, where he is currently pursuing the Ph.D. degree. He has published 3 books in the field of computer engineering, wired and wireless networks, and digital electronic in the Persian language. His research interests are in software defined networking, network function virtualization, 5G architecture and optimization.

Seyed Amin Hosseini Seno received his B.Sc. and M.Sc. degree in Computer Engineering from Ferdowsi University of Mashhad, Iran in 1990 and 1998 respectively and his Ph.D. from University Sains Malaysia, Penang, Malaysia in 010. Currently he is an Assistant Professor with the Department of computer engineering at the Ferdowsi University of Mashhad, Mashhad, Iran. His research interests include Wireless and Sensor Networks, Network protocols, QoS and Network Security.

Farzad Tashtarian received the B.S. degree in computer engineering from the Islamic Azad University of Mashhad, Mashhad, Iran, in 006, the M.S. degree in information technology from the Islamic Azad University of Qazvin, Qazvin, Iran, in 008, and the Ph.D. degree in computer engineering from the Ferdowsi University of Mashhad, Mashhad, Iran, in 013. Currently he is an Assistant Professor with the Department of Information Technology, Islamic Azad University of Mashhad and a member of the Young Researchers and Elite Club, Mashhad Branch, Islamic Azad University, Mashhad, Iran. From February 013 to October 013, he was a Visiting Scholar at the University of Missouri Kansas City (UMKC), MO, USA. His research interests include distributed optimization in networking aspects of wire and wireless communications, wireless sensor networks, multimedia networking, and software-defined networking.