J Supercomput (2017) 73:3796–3820
DOI 10.1007/s11227-017-1983-0

Cloudlet dynamic server selection policy for mobile task off-loading in mobile cloud computing using soft computing techniques

Shima Rashidi[1] · **Saeed Sharifian**[1]

Published online: 20 February 2017

Abstract Wide acceptance of mobile phones and their resource hungry applications have highlighted resource limitations of mobile devices. In this regard, cloud computing has provided mobile phones with unlimited resources in order to help them overcome their constraints and enable them to support wider range of applications; so, mobile devices can outsource their tasks to public or local clouds. To accommodate to exponential growth of requests, user requests should be distributed to different cloudlets and then transparently and dynamically redirected to the servers according to the latest network and server status. Therefore, finding the best place to off-load is vital and crucial to both functionality and performance of the system. However, accurate and timely parameters of network and servers' status are improbable to achieve, so the traditional algorithms cannot perform effectively and fully efficient. As a solution in this paper, an adaptive neuro-fuzzy inference system is proposed and trained to assign tasks to the servers efficiently. The trained system is robust to imprecise context information and is tolerable measurement noise and errors. We have considered improving both system performance and user quality of service parameters in this paper. Simulation results demonstrate that, compared with other server selection schemes, the proposed scheme can achieve higher resource utilization (utilization is a percentage of time that a server is busy doing something), provide better user-perceived quality of service, and efficiently deal with network dynamics. Simulation results show that our proposed algorithm excels over the compared works in terms of performance, at the best case about 30% and at the worst case about 8.93%.

✉ Saeed Sharifian
sharifian_s@aut.ac.ir

Shima Rashidi
shima.rashidi@aut.ac.ir

[1] Department of Electrical Engineering, Amirkabir University of Technology, Tehran 15914, Iran

Keywords Mobile cloud computing · Task scheduling · Off-loading · ANFIS

1 Introduction

Mobile phones are becoming more popular day by day, and their applications are developing more and more [1,2]. New coming mobile applications usually demand infinite battery power, high bandwidth, or high processing power [3], and development of these applications is hindered by the inability of mobile devices in execution of resource hungry applications; this inability is driven from their resource limitations. In the recent years, cloud computing was suggested as a novel solution for this problem so that the mobile phones can outsource their applications to clouds. Cloud computing (CC) can be defined as a computing service provider which is accessible through Internet [4]. Integration of mobile devices and cloud computing results in mobile cloud computing (MCC) is a promising technology which can help mobile devices to overcome their resource constraints and run various range of applications [5,6].

Public clouds which can provision service for mobile devices are located in coarse grained geographical regions. Because of their dispersion, mobile phones which use cloud services may tolerate delays due to their remoteness from the clouds. As a solution, cloudlets are established in fine grained regions to provide a faster and cheaper connection for mobile users [7]; a user can optionally decide where to off-load in order to improve its own profits and get its desired service faster. However, such high-handedly approaches may cause in using only one cloud or cloudlet as the resource provider, resulting in high latency and resource inadequacy. So, some server selection algorithms should be used in such systems. The selection method should consider the following factors: (i) user–server proximity in a MCC system; (ii) server load; or (iii) network connectivity between user and server. Intuitively, a good algorithm should choose a nearby, lightly loaded, and reachable server for a user request. In order to achieve mentioned goals in the server selection process, servers' load and network status are needed at any moment, but gaining such accurate data is unlikely and all we can have is estimated data; so, traditional algorithms may fail to suggest the best place. As a solution, we have proposed ANFIS (Adaptive Neuro-Fuzzy Inference System) which can act well with estimated data and is not vulnerable to data oscillations. Our goal is improving both system performance and user QoS (quality of service) parameters.

The remainder of this paper is as follows: Sect. 2 discusses related works and mentions our difference or priority upon them. In Sect. 3, an overall view of the system is given and different possible scenarios are discussed. In Sect. 4, the overall flow of the proposed algorithm is described and its parameters are introduced. Section 5 explains used ANFIS and formulates it. Results of the simulation with a brief explanation about each result are given in Sect. 6.

2 Related works

Mobile cloud computing and task allocation are developing concepts which have attracted much attention recently. A few surveys about mobile cloud computing, its challenges and applications can be found in [6,8,9]. According to these papers, task allocation algorithms can be categorized in many ways, such as central or distributed, dynamic or static, deterministic or context aware algorithms.

Task allocation algorithms can be central or distributed. In [10], two distributed dynamic load balancing algorithms, LBA and MELISA, in network environments are presented. These algorithms try to solve the problems caused by network delays, sharing, and job migration from one machine to another. They estimate completion time of jobs in processors and try to allocate tasks to resources in a way that response time of tasks is reduced. Authors in [33] have tried to reduce the energy consumption of networked data centers with a distributed and scalable scheduler. They have used an optimization algorithm in order to optimize the load provisioning process so both network devices and servers energy consumption will be decreased. Shojafar et al. [34] proposed a novel dynamic distributed resource allocation scheduler in mobile cloud computing environments for reducing energy consumption of both data centers and mobile connections. At the same time, it tries to meet quality of service requirements. The test results show its superiority over other algorithms. In [11], a central offline and a distributed online algorithms, used in heterogeneous mobile cloud computing environments (in which the configuration of the resource providers differs from each another), are presented. In this paper, besides response time of the executing tasks and their waiting time in the queues, cloudlet and user distance is also considered in the task allocation process. In [12], the authors proposed a probabilistic algorithm for online distributed scheduling, in which power consumption of the mobile devices and completion time of the tasks are reduced. Because of the resemblance of its assumptions with our proposed algorithm, it is chosen for comparison in the simulation scenarios. In [13], a distributed decision-making algorithm is proposed for off-loading to remote clouds. It uses game theory to find the optimal trade-off point between power consumption and network latency

Our problem can be modeled as a job scheduling problem which is mostly solved with static or dynamic algorithms. Static algorithms need to have prior knowledge of the system; they better answer in systems that do not have sharp fluctuations. There are a lot of deterministic algorithms for choosing the best place to off-load, but they usually consider only one or two specific parameters as their selection criteria. For example, a server might be chosen only because of its proximity to a user, its signal strength, or being less loaded. Despite the simplicity of these algorithms, inefficacy of them has been proven by practical applications [14]. An example of these algorithms is round-robin (RR) which is presented in [15] and is compared to our proposed algorithm in the simulation scenarios. This algorithm assigns tasks in a round and sequential manner. In the dynamic algorithms, decision is based on the current condition of the system. Therefore, status of all the nodes should be monitored constantly and the tasks should be allocated due to these statuses. These approaches are more flexible and can adapt to changes more easily; so they can have a better performance in dynamic and heterogeneous environments. In [16], in order to optimize management of the resources, mobile users are clustered based on their accessibility and movability. This clustering should be performed dynamically. Resources are assigned to each cluster based on its characteristics. Another dynamic algorithm which tries to reduce transmission and completion time of the tasks is proposed in [17]. This algorithm considers k resources for each task, using KNN (k-Nearest Neighbors) algorithm, and among those k resources, finds the most proper resource which reduces drop rate, power consumption, and completion time of the task.

Context aware algorithms operate real time and need updated context information constantly to make proper and aware decisions. The authors in [18] proposed a cost and energy aware algorithm for service provision in mobile clouds. Although updating context information has an overhead for the network [19], the context aware algorithms may have to update their information periodically [20]. Also, uncertainty in server load estimation and network status is a challenge in extracting precise context information which can be used for more proper decisions. In order to reduce algorithms' errors while facing imprecise context information, we suggest using ANFIS to make the algorithm more robust. ANFIS is used in many other fields such as prediction [21], modeling [22], load balancing [23], decision making [24].

3 System architecture introduction

In this part, the proposed distributed task allocation algorithm which can work in dynamic and heterogeneous environments (environments in which the configuration of the resource providers such as clouds and cloudlets differ from each another, so the response time of one task is different in each service provider), named Distributed Heterogeneous Task Allocation (DHTA), is proposed. In DHTA architecture, task allocation is online, adaptive and DHTA is designed in a way that mobile user mobility is considered. The cloudlets are connected together via Internet network and can send tasks to each other through the Internet backbone. Their connection with public clouds is also possible through Internet. As a result of Internet connection, user location would not matter anymore and then move after transmitting its task to its nearest cloudlet. Then, user's mobile device will go to the idle state till the task completion. When the user task is completed, a message about task completion is sent to the user. Results are uploaded on the cloudlet which is the nearest one to the user's new location and then sent to the user. In this condition, round-trip time (RTT) between cloudlets over Internet will be important and taken into account.

As shown in Fig. 1, the cloudlets are spread over different geographical areas and the users off-load their requests to their nearest cloudlet, named the host cloudlet, in their geographical zone which is determined by a circle with radius R. The task is actually off-loaded to the proxy of the cloudlet as a request to be processed for allocation; connection with the proxy of the cloudlet is possible via its access point through Wi-Fi. The proxy of the host cloudlet, after receiving the requests, studies network status, its own and other cloudlets' load and then makes a decision for offering the best place for processing each request. The best place to off-load is calculated in such a way that QoS parameters and system performance are optimized.

As it is known, bandwidth cost in 3G networks is higher than Wi-Fi [25] and also energy consumption of transferring data through 3G/4G network is higher than Wi-Fi [26,27]. Considering these two facts, three cases can happen in this system; in the first case, the host processes the request itself, so network delay, cost, and energy consumption would be minimized. This case occurs when forwarding the request to

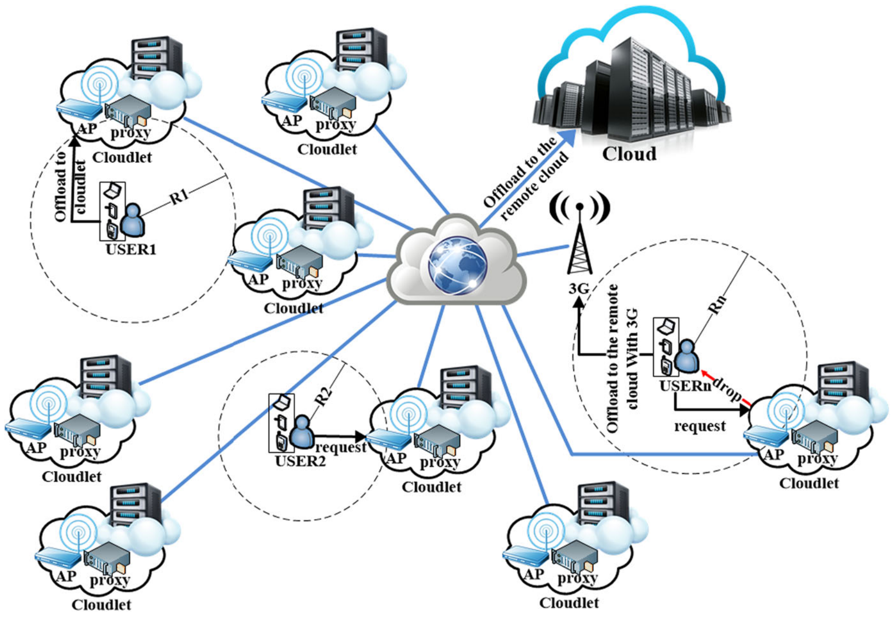


Fig. 1 System model for mentioned three cases

other cloudlets takes longer than waiting in the queue of the same cloudlet. In the next case, the host cloudlet off-loads the request to another cloudlet, named the executive cloudlet, or to the public cloud through Internet. In this case, in addition to server response time for processing the request, network delay caused by retransmission of the request should also be taken. In the last case, if there are not enough resources to process user requests or user waiting time is too much to bear, system will drop the request. To obtain optimal QoS parameters, this case should be avoided, since dropping the requests causes bad user experience and consequently reduces QoS parameters.

In the suggested system, it is assumed that there are n cloudlets and each cloudlet, $cloudlet_i, (i = 1, \ldots, n)$, receives m tasks in each time interval. Suppose that the request $R_j, (j = 1, \ldots, m)$ from the user $U_k, (k = 1, \ldots, u)$ is off-loaded to its nearest cloudlet, $cloudlet_i$. All the received requests are transmitted to the local cloud or other cloudlets or public cloud via the distributer in the host cloudlet. Since task transmission through Internet to other cloudlets results in time overhead for the system, it is preferred that $cloudlet_i$ executes the task itself in order to reduce cost, energy, and network delay. These decisions are made in decision-making unit which uses proposed AOTA (Adaptive Online Task Allocation) algorithm and its robust version, named FDM (Fuzzy-based Decision Maker). The purpose of this unit is selecting the server which is the closest one to the user and also has the least load. For an appropriate decision regarding how to find the best place to off-load, decision-maker unit needs to know information such as server parameters and network status among the cloudlets. RTT among the cloudlets and utilization of their servers are gathered in a repository in the proxy of the cloudlets. As shown in Fig. 2, decision-making unit in the host cloudlet

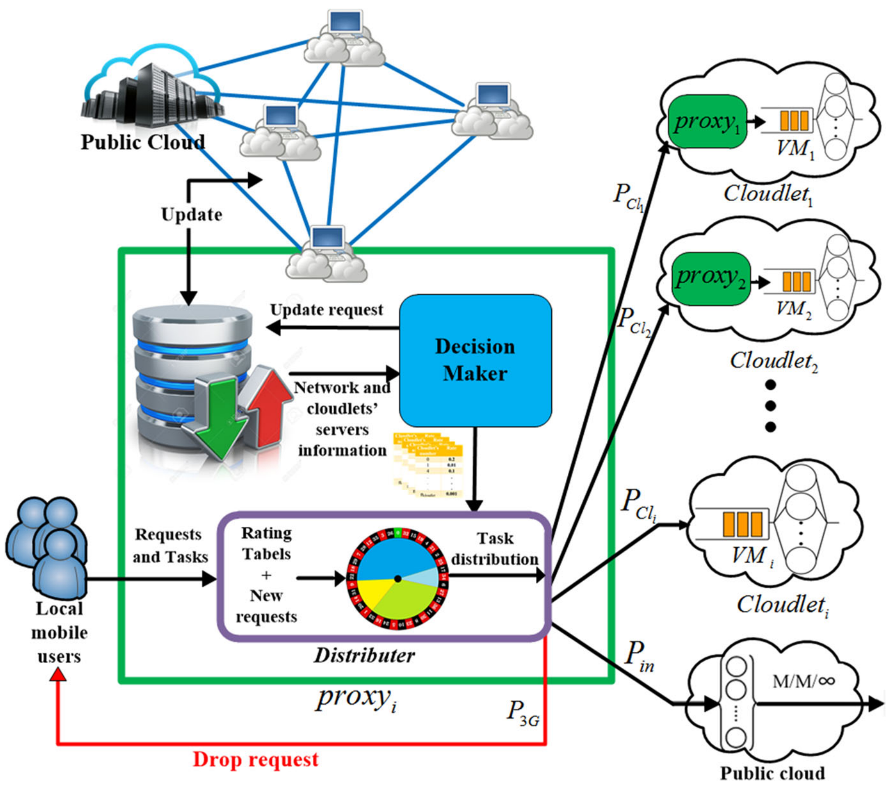


Fig. 2 Proposed off-loading mechanism

rates other cloudlets using this information considering its goal and produces a rating table in which the probability of outsourcing tasks to other cloudlets is inserted. Then, the distributer uses the produced rating table and distributes the tasks accordingly.

A deterministic decision maker should decide for each task where to off-load it. For this purpose, the decision maker should know the exact load of the cloudlets and exact RTTs, so it can send the task to the nearest and least loaded cloudlet. Since this information may not be accurate, the decision maker will not perform so well. Another problem is that during the data collection process, this information may be manipulated or their integrity may be distorted. So, maintaining the security and integrity of these information is of high importance. There are some routine protocols for data security purposes like SSL and TSL or even or even any VPN. Also, the integrity of messages is guaranteed by hashing mechanism. These standards require asymmetric cryptography like RSA and certificate and key management infrastructure. So, we can use such approaches to insure the security of data collection and integration in our system [35]. After the data collection stage, the obtained context information, used for decision making, is updated via Internet periodically using the mentioned protocols. Then, they are used at the start of a period to make a decision for all incoming requests in that period. So, this information may be inaccurate as they belong to the previous period. As the network traffic is dependent on various parameters and is constantly changing, it cannot be measured precisely. Also, network requests are not completely predictable, because of users' mobility, so the network traffic characteristics cannot correctly be estimated. Therefore, measuring and updating accurate network delay between different servers is not possible. On the other hand, when the number of servers in a network is high, server load estimation is also almost impossible, since the number of requests is constantly changing. According to the subjects that were told, estimation of immediate network parameters and servers' loads is very difficult. Given that the proposed algorithm should operate online, a solution is presented below.

In every periodic time interval, depending on the performance of the system, the average number of the requests and servers' loads in that time slot is obtained. It is expected that in the next period, the system shows the same treatment and its parameters do not change so much. Therefore, the obtained values in the prior period would be used in the next period; then, the rating table for each cloudlet would be updated. But measurements are not accurate and may have some deviations; so a fuzzy system is suggested. The proposed fuzzy system has multiple inputs and also is noise resistant, so it can perform well with inconsistent input values.

4 Formulation of the proposed algorithm

In this section, the problem is formulized; then, the assumptions and used parameters are mentioned. Using the rating tables produced in the decision-making unit, tasks are sent to *cloudlet*$_i$ with probability P_{CL_i}. For modeling the system and better predicting its behaviors, cloudlets are modeled as M/M/Vs/Qmax systems. In this model, according to Kendall notation, incoming request rate is a Poisson procedure and the modeled cloudlet has Vs servers and one queue. The cloudlet can have maximum number of Qmax clients in its queue, and extra tasks should be sent to other cloudlets or dropped. Service rate of the servers is an exponential distribution with mean μ. Public cloud is modeled as a M/M/∞ queue. This queue model also has Poisson arrival rate and exponential service rate. Public clouds have somewhat infinite sources in comparison with cloudlets, so its queue and number of servers are considered infinite to bold this difference. Therefore, response time of the executing tasks in a public cloud is almost zero and completion time is only proportional to the transmission time of sending tasks to the public cloud via Internet (μ_{in}). Probability of sending a task to the public cloud is P_{in}. The tasks are dropped with probability of P_{3G} and should be off-loaded to public cloud via 3G connection; it takes $\mu_{in} + \mu_{3G}$. In conclusion, completion time of all tasks in the system can be calculated with (1).

$$T_f = P_{3\mathrm{G}}(\mu_{3\mathrm{G}} + \mu_{\mathrm{in}}) + P_{\mathrm{in}}\mu_{\mathrm{in}} + P_{\mathrm{Cl}}T_{\mathrm{total}} \tag{1}$$

in which T_{total} is the mean completion time of the allocated tasks to the cloudlets.

In order to formulize this problem, host cloudlets are shown with index i and executive cloudlets are specified with index ($j = 1, \ldots, n$). i and j can be equal when a host cloudlet is the executive itself. Each incoming request to host cloudlet$_i$ is specified with cloudlet's index (i). In order to model allocating tasks to cloudlets, a $n \times n$ matrix, named S, is considered. In each period, every time that an incoming request to host cloudlet$_i$ is outsourced to executive cloudlet$_j$, corresponding element, S_{ij} is incremented. S is set to zero at the start of each period. A $1 \times n$ array, named alloc, is considered to save the number of allocated tasks to each cloudlet. Aggregate number of tasks outsourced to executive cloudlet$_j$ and its incoming tasks which itself executes are inserted in $alloc_j$. The alloc array is used for calculating the utilization of cloudlets and is used in our proposed algorithm as the context information for the next period. Completion time of each incoming task to executive cloudlet, Cl_j, from host cloudlet$_{\mathrm{i}}$ is shown with T_{ij}^{c} and can be calculated with (2).

$$T_{ij}^{c} = T_{ij}^{txr} + T_{j}^{r} \tag{2}$$

T_j^r is response time of executing the task in Cl_j and T_{ij}^{txr} is transmission time of sending the task from Cl_i to Cl_j. If i equals j, T_{ij}^{txr} is zero. It should be noticed that the size of all tasks is considered the same in this article, so transmission time is proportional to round-trip time of sending the task. As calculated in (3), response time of executing a task in executive cloudlet$_{\mathrm{j}}$ is a summation of service time of cloudlet's servers and the time taken for the task waiting in the cloudlet's queue.

$$T_j^r = T_j^s + T_j^w \tag{3}$$

in which T_j^s is service time of cloudlet's servers and T_j^w is the time taken for the task waiting in the cloudlet's queue. Completion time of all the tasks to all the cloudlets is calculated with aggregating completion time of each task. It can be calculated using (4).

$$T_{\text{Total}} = \sum_{i=1}^{n} \sum_{j=1}^{n} (T_{ij}^{c} \times S_{ij}) \tag{4}$$

One of the proposed algorithm's main goals is reducing total completion time in order to meet QoS conditions. This condition is written in (5).

$$\min(T_{\text{Total}}), \text{ s.t. } \sum_{j=1}^{n} \text{alloc}_j < Q\max_j \tag{5}$$

in which $Q\max_j$ is the queue length for cloudlet$_{\mathrm{j}}$. If the condition in (5) is not met, second scenario will occur and extra tasks will be sent to public cloud. According to (6), this scenario has financial costs for the user.

$$\sum_{j=1}^{n} \text{alloc}_j > Q\max_j \rightarrow \text{Cost}_C = \alpha_c N_c \tag{6}$$

in which $Cost_C$ is total expenses for system to off-load to public cloud. α_c and N_c are cost of a service in public cloud and number of tasks transmitted to public cloud by cloudlets, respectively. The bandwidth dedicated to cloudlets is limited, and number of tasks which can be sent to public cloud by each cloudlet is limited to N_{thr}. Therefore, if N_c exceeds N_{thr}, extra requests will be dropped. In this way, N_d requests will be off-loaded to public cloud via 3G connection. This causes extra financial and time costs for the dropped users (7).

$$N_c > N_{thr} \rightarrow \text{Cost}_d = (\alpha_{3G} + \alpha_c)N_d \tag{7}$$

where $\alpha_{3\text{G}}$ is cost of task transmission via 3G connection.

5 Procedure explanation of the proposed algorithm

Our proposed algorithm considers the aforementioned problem formulation and acts online and distributed. It is adaptive and can perform in heterogeneous environments as well. Distributed task allocation feature can be explained as followed. All the nodes are aware of each other's information, and each cloudlet decides for its own incoming requests, whether to process them or outsource them. In such a system, the cloudlets broadcast just their utilization and they are not aware of each other's incoming tasks, so the proposed algorithm is distributed. In the proposed system, all the cloudlets receive different requests in each period simultaneously and all the host cloudlets rank their hypothetical executive clouded based on the shared context information. In this situation, if all cloudlets choose a cloudlet with lower response time in a greedy manner that cloudlet will get overloaded and its response time will increase. To solve this problem, AOTA has considered RTT as well as cloudlets' utilization and also adopted a probabilistic approach. As the result, the cloudlets with higher probability get higher chance to be chosen and the cloudlets with lower probability have also a little chance to be chosen. The probability of sending a task from host cloudlet (Cl_i) to executive cloudlet (Cl_j) can be calculated using (8) and it should get inserted in each cloudlet's rating table.

$$P_{\text{Cl}_i \rightarrow \text{Cl}_j} = \frac{\frac{\alpha}{\alpha \times T_j^w + (1-\alpha) \times RTT_{ij}}}{\sum \frac{1}{T_j^w + \text{RTT}_{ij}}}, \qquad \alpha \in [0, 1] \tag{8}$$

in which α is the adaptability coefficient which puts weigh on either load balancing or transfer time reduction by the system administrator. This equation helps to have load balancing while reducing transfer time. If α is set to 1, the algorithm just focuses on load balancing, and transfer time reduction is not considered for task allocation. If α is set to 0, the algorithm minimizes transfer time without load balancing.

In the proxy of every cloudlet, there is a decision-making unit which decides for input requests where to off-load. This unit needs precise context information to make a proper decision. Since updating these information causes time overhead, decision-making unit cannot access context information at any moment instantly. So, it has to use previous period's information to assign tasks in current period. Since the information is old and the system cannot decide based on its current status, the algorithm will face some errors. In order to solve this problem, fuzzy inference system is suggested,

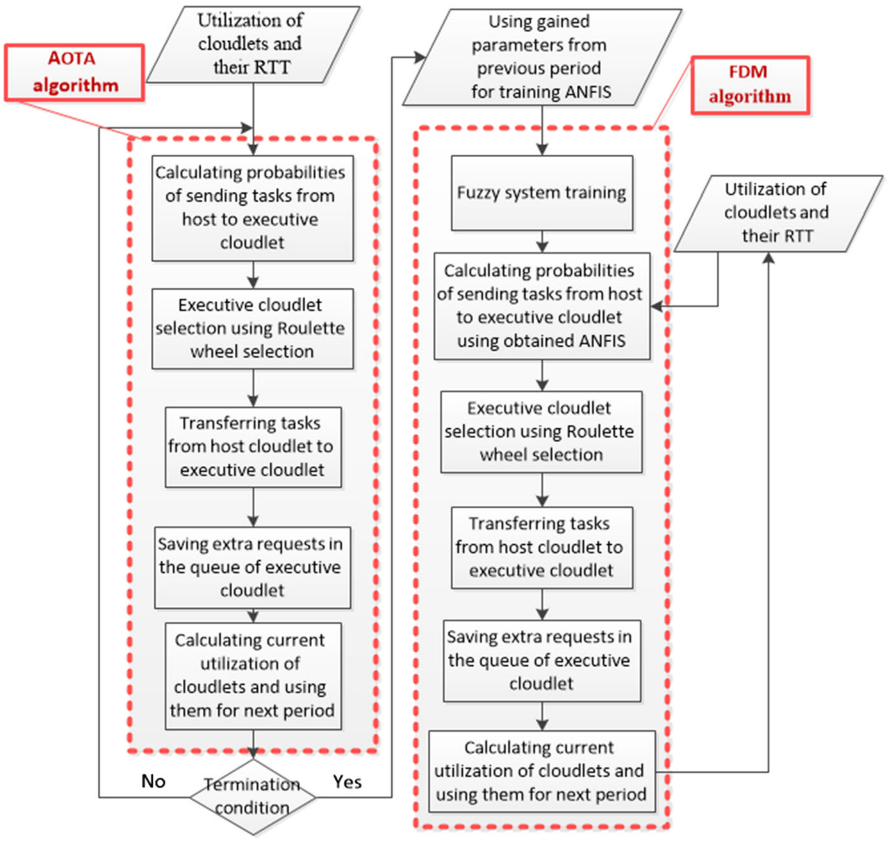
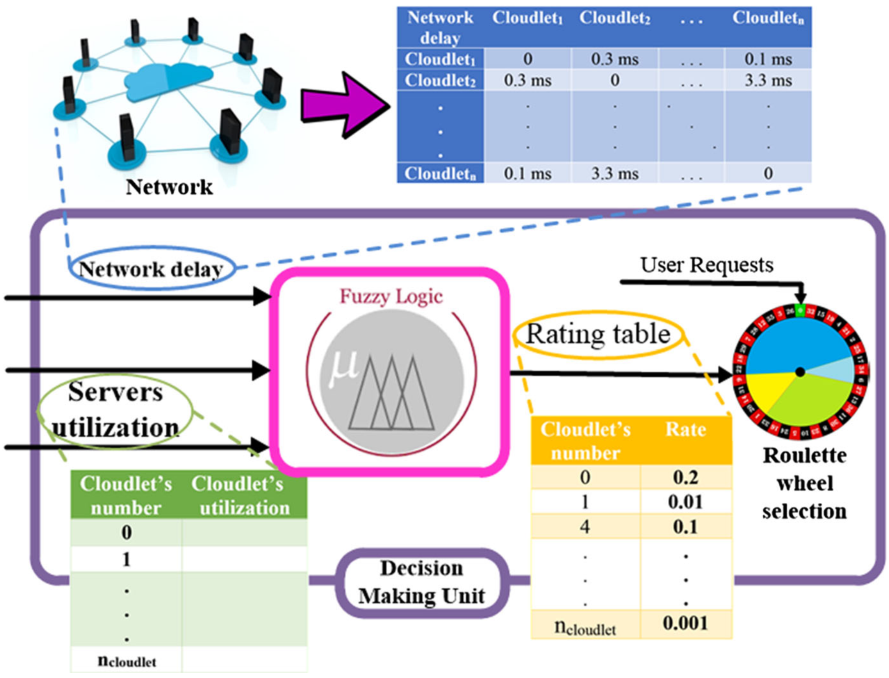

AOTA algorithm
Utilization of cloudlets and their RTT
Calculating probabilities of sending tasks from host to executive cloudlet
Executive cloudlet selection using Roulette wheel selection
Transferring tasks from host cloudlet to executive cloudlet
Saving extra requests in the queue of executive cloudlet
Calculating current utilization of cloudlets and using them for next period
No
Termination condition
Yes
Using gained parameters from previous period for training ANFIS
FDM algorithm
Fuzzy system training
Calculating probabilities of sending tasks from host to executive cloudlet using obtained ANFIS
Utilization of cloudlets and their RTT
Executive cloudlet selection using Roulette wheel selection
Transferring tasks from host cloudlet to executive cloudlet
Saving extra requests in the queue of executive cloudlet
Calculating current utilization of cloudlets and using them for next period

Fig. 3 Flowchart of FDM algorithm

so that the system will be robust to the uncertainty of input control parameters. As the result, in the decision-making unit there is a fuzzy inference engine. The fuzzy system inputs are utilization of cloudlets' servers and the RTT of sending a task from cloudlet$_i$ (host) to cloudlet$_j$ (executive), which is shown with RTT$_{ij}$. The output of the fuzzy system is a rating table for each server to allocate its received tasks accordingly. Thus, the rating table would be updated for each time slot and each request would be allocated to the cloudlet with higher rate more likely.

The dataset that is used in this algorithm is gathered with the use of aforementioned algorithm, AOTA, which objective is minimizing the servers' response time and network latency of the system. Making it robust and better, AOTA is developed to FDM by integrating it with a fuzzy inference system. One of the most important benefits of FDM is that it can be trained with any other task allocation algorithm with any other objective. This algorithm should be implemented for each server individually in each period to update the rating table for new requests. Flowchart of AOTA and FDM algorithms is illustrated in Fig. 3.

The proposed task allocation algorithm is performed in the proxy of each host cloudlet and decides only for host cloudlet's arrival requests. Utilization of cloudlets



Network delay	Cloudlet$_1$	Cloudlet$_2$	...	Cloudlet$_n$
Cloudlet$_1$	0	0.3 ms	...	0.1 ms
Cloudlet$_2$	0.3 ms	0	...	3.3 ms
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
Cloudlet$_n$	0.1 ms	3.3 ms	...	0

Cloudlet's number	Cloudlet's utilization
0	
1	
.	
.	
.	
n$_{cloudlet}$	

Cloudlet's number	Rate
0	0.2
1	0.01
4	0.1
.	.
.	.
.	.
n$_{cloudlet}$	0.001

Fig. 4 Decision-maker unit used in the proxies of cloudlets

and their RTT is the inputs of AOTA algorithm, and this information belongs to previous period. Using (8), the probability of sending a task to executive cloudlets is calculated and inserted in the host's rating table. Executive cloudlet of each task is determined using the rating table and roulette wheel selection. At the end of the period, utilization of cloudlets will be calculated and network status is updated. AOTA is executed on a standard workload offline, and the results are saved to be used as the training data for FDM algorithm. FDM has a similar performance like AOTA, although FDM has more certainty and status adaptability.

Suggested decision-making unit, using FDM algorithm, is illustrated in Fig. 4. Designed fuzzy inference system uses a series of knowledge-based rules and a fuzzy inference engine to estimate the probability of sending a task from host cloudlets to executive ones. RTT of sending a task between cloudlets and servers' utilization is two inputs of fuzzy inference engine. So $\mathrm{RTT}_{ij}, i, j = 1, \ldots, n$, and $\rho_j, j = 1, \ldots, n$, are linguistic variables and P_{ij} (probability of sending a task from cloudlet$_i$ to cloudlet$_j$) is the output of the fuzzy inference engine. These probabilities are inserted in cloudlet$_i$'s rating table. The produced rating tables go to the Roulette wheel selection, and final decision for each task is the output of the decision-making unit.

5.1 Used ANFIS in the proposed system

"A fuzzy set is a class of objects with a continuum of grades of membership," Zadeh proposed the fuzzy set theory for the first time [28]. Fuzzy logic is the process of finding a relationship between given fuzzy inputs and outputs. In this section, we tune an initial FIS, using ANFIS. ANFIS is proposed in early 1990 and is an adaptive neuro-fuzzy inference system which works based on Takagi–Sugeno (TS) [29]. This method combines both learning features of neural networks and fuzzy logic aspects, so we can efficiently train a system to work with fuzzy inputs. Its inference system is adaptive with If–Then rules and provides the learning ability for estimating nonlinear functions. At first, a dataset with input parameters and target measurements should be loaded to create initial FIS. The initial FIS is created using FCM method, which extracts a set of proper rules regarding input and target data behavior. The generated FIS is of Sugeno type with Gaussian membership functions.

Every fuzzy inference system has some specific parameters such as type, defuzzification method, input, output, rule. Output and input membership functions can be defined of any distribution, so it can be trained by changing these parameters in a proper way. So as the next step, ANFIS is used to calculate parameters of the membership functions of fuzzy antecedent parameters, and linear consequent parameters of fuzzy rules for Takagi–Sugeno [30] FIS. Takagi–Sugeno (TS) is used as a basic fuzzy inference system for ANFIS [31]. Takagi–Sugeno can model many nonlinear systems and be used for approximating them. Each rule in TS can be a linear function or crisp number, as shown in (9).

$$R_i : \text{IF}\, x \text{ is } A_i \text{ THEN } y_i = a^T x + b_i, \quad i = 1, 2, \ldots, M \text{ and } 0 < R_i < 1 \tag{9}$$

in which X, the inputs, is the antecedent and y_i is the consequent of the i'th rule. A_i is x' membership function; a_i and b_i are consequent parameters which can be regulated in order to improve model's behavior. The TS antecedent usually uses and-conjunction.

$$\mu_i(x) = \prod_{j=1}^{p} \mu_{ij}(x_j) \tag{10}$$

which is gained by multiplying membership degrees of x_j in A_i. So, the total output for the input x is computed by aggregating the individual rules contributions:

$$y = \frac{\sum_r \overline{y_r}.\exp\left(-\frac{1}{2}\sum_r\left(\frac{x_i - m_{ri}}{\sigma}\right)^2\right)}{\sum_r \exp\left(-\frac{1}{2}\sum_i\left(\frac{x_i - m_{ri}}{\sigma}\right)^2\right)} \tag{11}$$

in which x_i $(i = 1, ..., n)$ is inputs of the fuzzy system, r is the representative of different rules and $\overline{y_r}$ is the consequent of the r'th rule. σ and m_{ri} are parameters of membership functions for each input. These parameters can be regulated in order to improve system approximation.

In Fig. 5, the used ANFIS is illustrated. ANFIS is based on Takagi–Sugeno fuzzy inference system, so its rules are as follows:

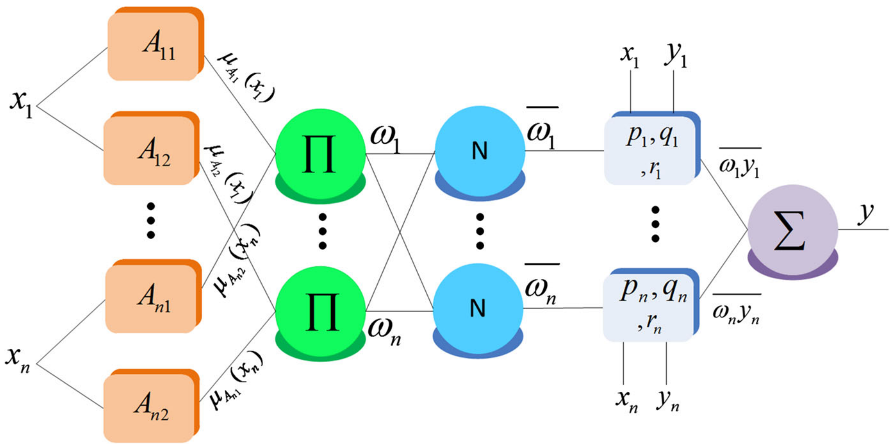


Fig. 5 Used ANFIS algorithm

$$R_1 : \text{IF } x_1 \text{ is } A_{11} \text{ and } x_2 \text{ is } A_{21} \text{ Then } y = p_1x_1 + q_1x_2 + r_1 \tag{12}$$

$$R_2 : \text{IF } x_1 \text{ is } A_{12} \text{ and } x_2 \text{ is } A_{22} \text{ Then } y = p_2x_1 + q_2x_2 + r_2 \tag{13}$$

$$\Rightarrow \text{R}_r : \text{IF } x_i \text{ is } A_{ij} \text{ Then } y = p_jx_1 + q_jx_2 + \cdots + r_j \quad i = 1, \ldots, n \text{ and } j = 1, \ldots, m \tag{14}$$

As it is shown, each n input can be defined with m membership functions in the first layer and $\mu_{nm}(x_n)$ is the membership degree of x_n in A_{nm} fuzzy set. Layer two is the T.norm creating fuzzy rules that assigns the minimum membership degree of x_i in its membership functions as w_r. So w_r for each rule (used in (11)) is achieved and will be normalized in the next layer.

$$\overline{w_i} = \frac{w_i}{\sum_{i=1}^{n} w_i} \tag{15}$$

So, in the next layer weighted outputs will be calculated and at last the output will be obtained from (16).

$$y = \frac{\sum_r \overline{y_r} w_r}{\sum_r w_r} = \sum_{i=1}^{n} y_i \overline{w_i} \tag{16}$$

For a better answer, the membership function parameters are extracted from the FIS resulted from the previous step and are fed into an optimization algorithm. For this problem, AI algorithms are used since the number of these parameters are very high, resulting in a big solution space. Cost function would be obtained from (18), in order to reduce FIS error.

$$e_i = t_i - y_i \tag{17}$$

Table 1 Parameters that are used in the FDM algorithm

Parameter	Value
Number of cloudlets	20
Period time used for updating context information (s)	5
Mean of service rate (μ) (req/s)	0.98
Mean of number of servers in each cloudlet (VS_j)	2.55
Maximum number of cloudlets (Qmax)	5
Maximum number of tasks allowed to off-load to public cloud (N_{thr})	4
Adaptability coefficient (α)	0.4

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} e_i^2 \tag{18}$$

where t is the train data targets which is desired and y is the output of the FIS for the train data inputs.

6 Evaluation of the proposed method

In this section, simulation results are illustrated and the proposed algorithm is compared to the state of art.

6.1 Evaluation setup and comparison scheme

In order to evaluate performance of the proposed algorithm properly, a scenario is simulated as follows: There is an urban area in which 20 cloudlets are distributed randomly. Mean delay of transferring a 20Mb packet from a cloudlet to another one is about 7/568 s. The used parameters in the algorithm are given in Table 1. The values of these parameters are gained practically through a large number of trial and error experiments. They are set to the values with which the algorithm shows its best performance.

All the cloudlets are connected through the Internet backbone. Noting Fig. 6, RTT for connection between cloudlets can be calculated using (19).

$$\text{RTT}_{ij} = \text{RTT1} + \text{RTT2} + \text{RTT3} \tag{19}$$

in which RTT1 is the network delay of cloudlet_a from the metro ring, RTT2 is a very short delay time on the metro ring and RTT3 is the network delay of metro ring from cloudlet_b.

Average of incoming requests to all the cloudlets in 500 seconds (100 Time Steps), taken from two weeks of HTTP logs from a busy Internet service provider WWW server, CLARKNET [32], is shown in Fig. 7.

Parameters which are studied in this article are as follows:

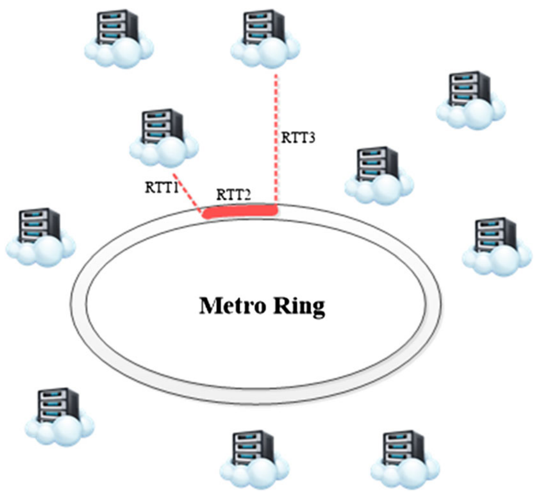


Fig. 6 RTT for cloudlets' connection

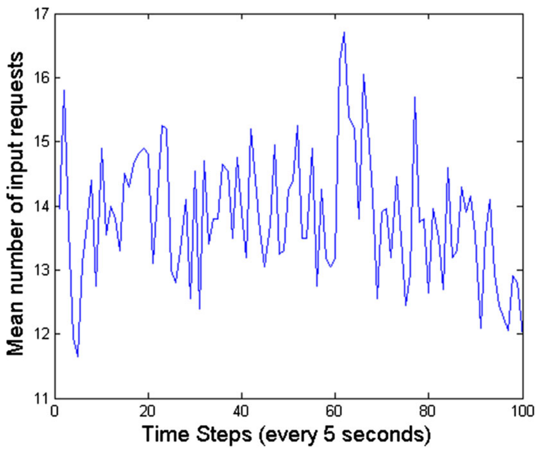


Fig. 7 Average of incoming requests to all cloudlets

- *Mean transfer time* Host cloudlet$_i$ sends some of its tasks to other cloudlets in a specific transfer time ($\sum_j T_{ij}^{txr}$) which is proportional to $\sum_j RTT_{ij}$. So, mean transfer time in each period can be calculated with (20).

$$\mathrm{mean}(T^{txr}) = \frac{1}{n}\sum_i \sum_j T_{ij}^{txr} \times S_{ij} \tag{20}$$

in which n is the total number of cloudlets and S_{ij} is the assignment matrix.

- *Utilization CV* Utilization shows that to what percent the cloudlets are busy. Utilization of each executive cloudlet j (ρ_j) is calculated by (21).

$$\rho_j = \frac{alloc_j}{Vs_j\mu_j} \tag{21}$$

in which $alloc_j$ is the number of requests that cloudlet$_j$ should execute in each period. Vs_j and μ_j are number of servers and service time of cloudlet$_j$, respectively. CV is coefficient of variation which can be calculated with (22).

$$E(x) = \mu = \frac{1}{n}\sum_{i=1}^{n} x_i \rightarrow \sigma = \sqrt{E[x^2] - (E[x])^2} \rightarrow \mathrm{CV} = \frac{\sigma}{\mu} \qquad (22)$$

in which $E(x)$ is the average of variant x and σ is the standard deviation. Utilization CV of cloudlets is a proper criterion for load balancing evaluation. The more little its value is the more balanced the loads of cloudlets are. Little utilization CV represents that the tasks are distributed more uniformly among the cloudlets' servers and the cloudlets are more equivalently utilized.

- *Mean completion time* Mean completion time can be calculated with (23) using equation mentioned in (2) in each period.

$$T_{\text{Total}} = \frac{1}{\sum \text{alloc}} \sum_{j=i}^{n} \sum_{j} (T_{ij}^{c} \times S_{ij}) \qquad (23)$$

- *Summation of drop requests* Summation of dropped requests of all cloudlets in each period is considered and evaluated as a parameter for QOS of the system.

6.2 Evaluation results

In this section, evaluation results are given and discussed. At first, the performance of the algorithm is verified and then it is compared with the state of art.

In order to verify the performance of rating tables (Rtable) in FDM algorithm, trend of probabilities of sending tasks to other cloudlets for 4 host cloudlets as samples in 100 time steps is shown in Fig. 8. System prefers that the host cloudlet processes its incoming request itself, so the task's completion time would be minimized. So, as shown in Fig. 8, the queue of host cloudlets with less servers (Fig. 8a, c with two servers) get full faster and the probability of executing incoming tasks themselves reduces (blue and yellow graph in Fig. 8a, c, respectively). But host cloudlets with more servers (Fig. 8b, d with 3 and 5 servers, respectively) prefer to execute the incoming task themselves most of the times, so the probability for them doing the task is high in most of the time steps (red and green graphs in Fig. 8b, d, respectively).

Another parameter which is considered in the proposed algorithm and studied here is the queue length. If the number of tasks in the queue of cloudlets reaches the queue length, extra requests will be dropped. Queue length is not specific and is determined by the cloudlet's owner. So, the proposed algorithm should adapt with any queue length. Increasing queue length results in reduction of dropped requests since more requests can wait in the queue and the queue will not overflow quickly. The simulation results of AOTA's performance with different queue lengths are given in Table 2.

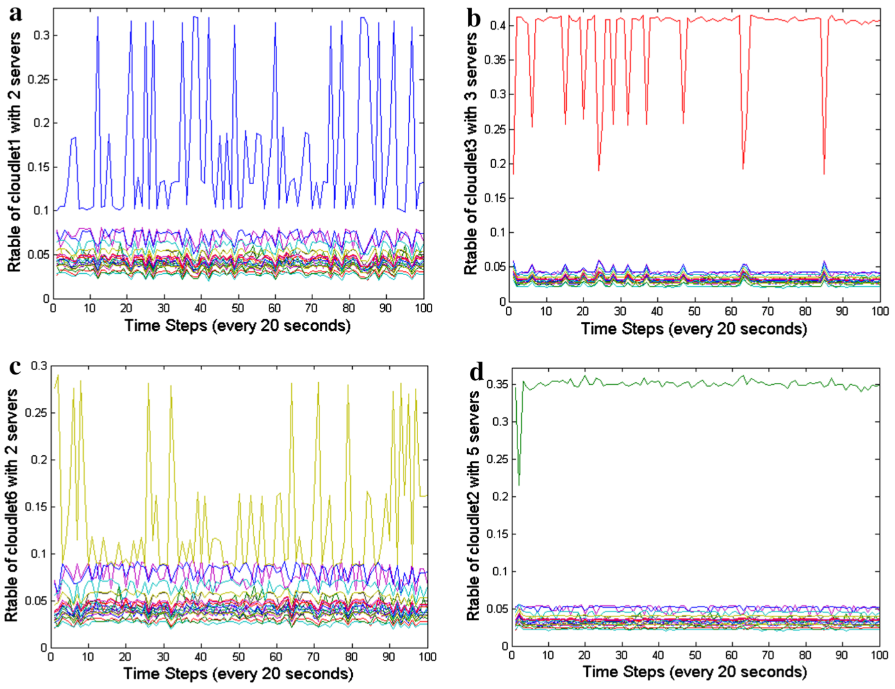


Fig. 8 Rating tables for host cloudlets **a** rating table for a host cloudlet with 2 servers, **b** rating table for a host cloudlet with 3 servers, **c** rating table for a host cloudlet with 2 servers, **d** rating table for a host cloudlet with 5 servers

Table 2 Average and standard deviation of "summation of dropped requests"

	$Q\max = 5$	$Q\max = 8$	$Q\max = 10$	$Q\max = 15$	$Q\max = 20$
Average	30.8430	30.7525	28.9675	27.0340	25.9875
SD	11.7482	11.6351	11.7476	11.2450	11.5707

As given in Table 2, with the increase in queue length from 5 to 20 requests, average of sum of dropped requests is decreased. If we increase aggregate number of servers of all cloudlets to 79, as shown in Fig. 9, number of dropped requests reduces significantly.

Number of servers' accretion cause more requests being served simultaneously, so the number of tasks waiting in the queues will reduce. As a result, it will take longer for the queues to get full and so under the condition in which queue length is 20, drop request rate gets zero (Fig. 9).

In order to verify adaptability of the proposed algorithm, value of parameter α is changed from 0.1 (more focus on task completion time reduction) to 0.9 (more focus on load balancing of cloudlets) and its results on utilization CV of the cloudlets and mean transfer time of all tasks being send to other cloudlets are shown in Figs. 10 and 11. Average and standard deviation of these graphs is given in Table 3. Utilization CV

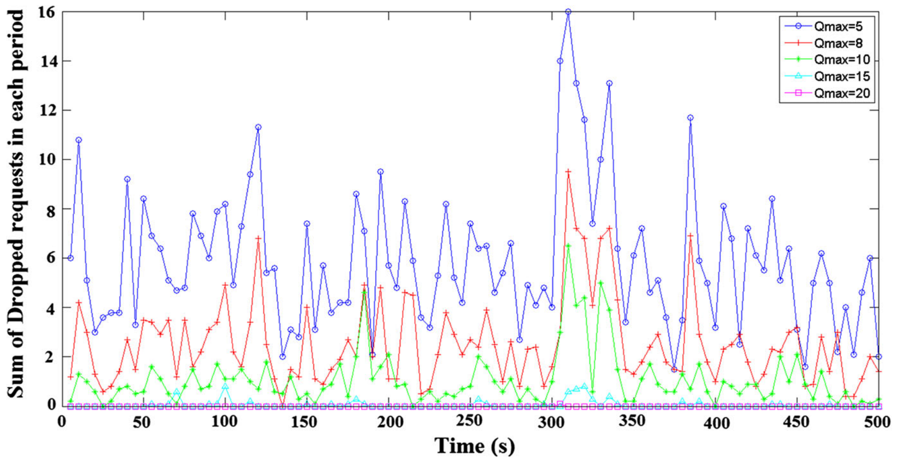


Fig. 9 Summation of dropped requests of all cloudlets with different queue lengths

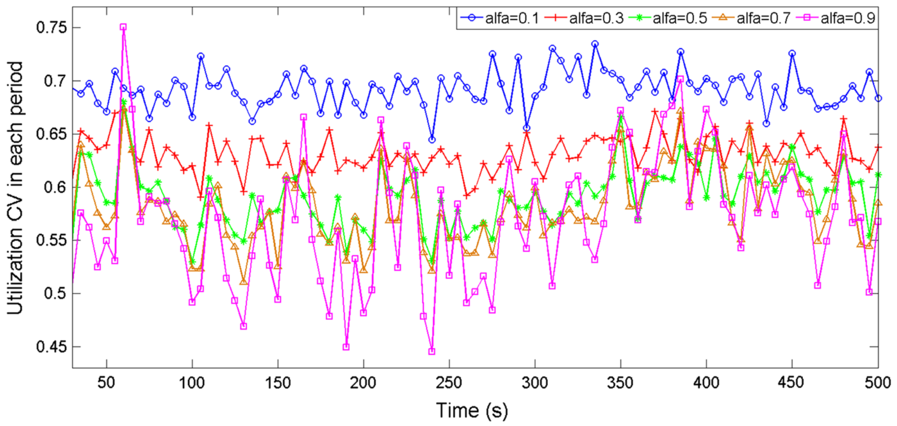


Fig. 10 Utilization CV of cloudlets in terms of α variation in each period

and mean transmission time are studied as representatives of load balancing and mean completion time, respectively.

Regarding Figs. 10 and 11 and due to α's nature in (8), it is verified that with the increase of α, utilization CV is reduced and mean transmission time is increased.

With the above-mentioned experiments, well performance of AOTA algorithms is proved. In the next step in order to reduce the error of online execution and using last period's context information, an ANFIS is trained to simulate AOTA and FDM algorithms. The results of training ANFIS are shown in Figs. 12 and 13. Two thousand and three hundred samples are extracted from offline execution of AOTA as the train data. These data include two input data (executive cloudlet's utilization and RTT of host and executive cloudlet in the network) and one output (probability of sending the task from host cloudlet to executive cloudlet). Input data of the system are shuffled, and then, 70% of it is used as train data and residual 30% is used as test data for the

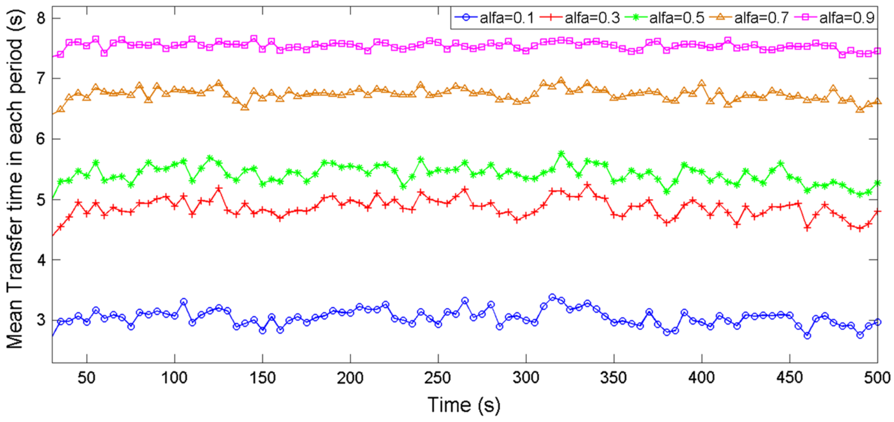


Fig. 11 Mean transfer time in terms of α variation in each period

Table 3 Average and standard deviation of "utilization CV and mean transfer time in terms of α variation"

	α = 0.1	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 09$
Average (Fig. 11)	0.6927	0.6324	0.5937	0.5856	0.5753
SD (Fig. 11)	0.0182	0.0180	0.0312	0.0390	0.0673
Average (Fig. 12)	3.0821	4.8814	5.4293	6.74	7.5378
SD (Fig. 12)	0.4701	0.3179	0.2616	0.143	0.0676

ANFIS. To evaluate the performance of training process, two factors, Mean Squared Error (MSE) and Root Mean Square Error (RMSE) are used. Selection of the type of membership function has a giant impression on the system performance and ease of system compliance. The most commonly used membership functions are Gaussian and triangular functions. In this article, we have chosen the Gaussian functions because they can better model the nature of the network and load conditions of the cloudlets in the proposed architecture.

RMSE of the generated Fuzzy Inference System is only about 0.014 for training data and 0/01472 for the test data. So, the system has been trained well to follow AOTA algorithm.

After all the above steps, we will have a trained system for task allocation in dynamic, distributed and heterogeneous mobile cloud computing environments. The system takes utilization rate of cloudlets and RTT of them in the Internet network from the previous period, and gives the proper probability of outsourcing incoming tasks for each host cloudlet. FDM is compared with one other algorithms from previous works done in [12], round-robin (RR) algorithm (as a central algorithm) and a probabilistic algorithm (Ra) in order to evaluate its performance in comparison with the state of art. [12] is a probabilistic algorithm for online distributed scheduling, in which power consumption of the mobile devices and completion time of the tasks are reduced.

In the first experiment, utilization CV of cloudlets are compared in order to study algorithms' ability in balancing the load of cloudlets. Utilization CV is an appropriate

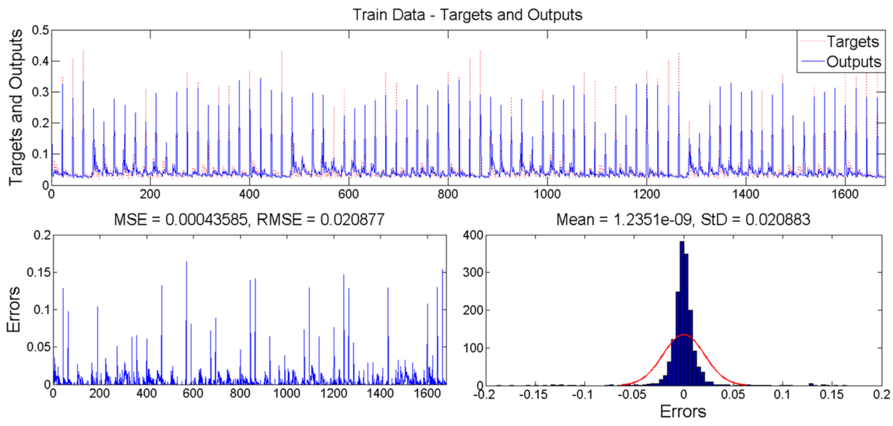


Fig. 12 ANFIS output for train data

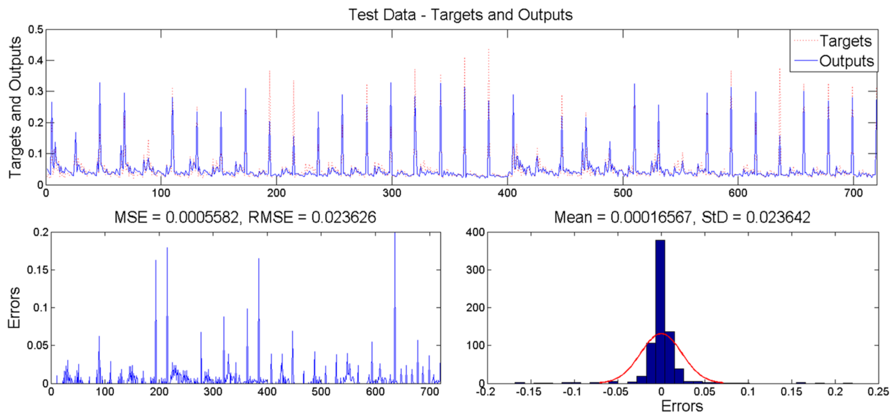


Fig. 13 ANFIS output for test data

criterion for load balancing verification and little values of it shows more balanced servers and vice versa. Figure 14 shows that FDM algorithm decreases utilization CV more because of its appropriate cost function and probability criteria. Average and standard deviation of Utilization CV graph is inserted in Table 4.

In the second experiment, drop rate of the system using different algorithms is studied. Reduction of drop rate results in client pleasure and refinement in quality of service parameters. Experiment results are shown in Fig. 15. As it can be seen, FDM acts better in drop rate reduction since it balances loads of cloudlets more equivalently. Better distribution of tasks among cloudlets causes lower drop rate. Matching Fig. 14 and 15, it can be found out that in the period intervals 50–100 and 300–350, number of dropped requests is increased because of unbalanced loads in the cloudlets. Average and standard deviation of some of dropped requests graph is given in Table 5.

In the next experiment mean Transfer time using different algorithms is studied. Results are shown in Fig. 16 and Table 6 from which it can be concluded that FDM has

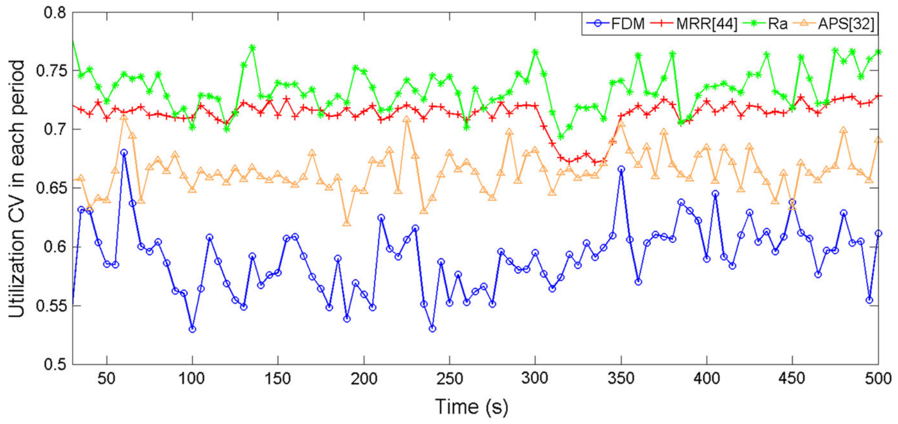


Fig. 14 Utilization CV of cloudlets using different algorithms in each period

Table 4 Average and standard deviation of "utilization CV of cloudlets" over time

	FDM	RR	Ra	ASP [12]
Average	0.5937	0.7122	0.7340	0.6657
SD	0.0312	0.0155	0.0183	0.0194

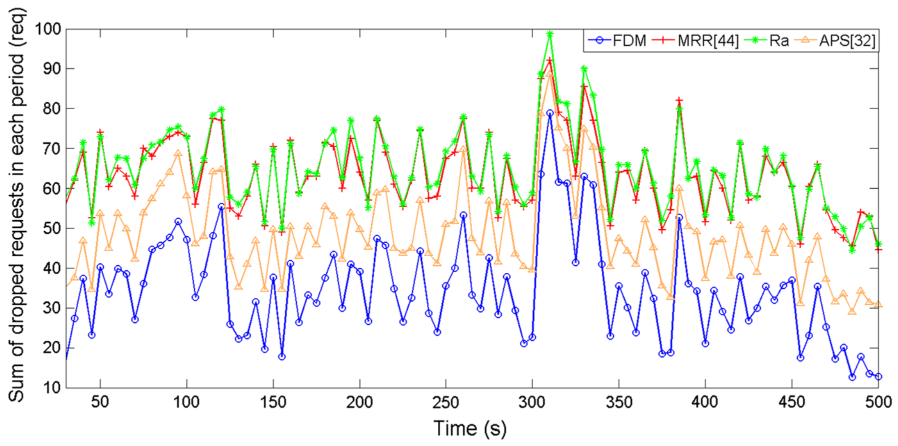


Fig. 15 Summation of dropped requests using different algorithms in each period

Table 5 Average and standard deviation of "Summation of dropped requests" over time

	FDM	RR	Ra	ASP [12]
Average	34.1250	62.7650	64.1880	47.9240
SD	12.7807	10.9280	11.0975	11.9622

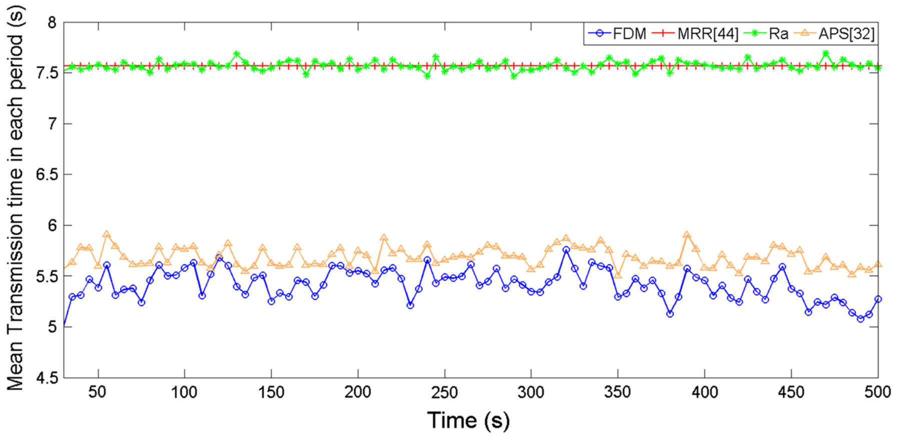


Fig. 16 Mean transmission time using different algorithms in each period

Table 6 Average and standard deviation of "Mean Transmission Time" over time

	FDM	RR	Ra	ASP [12]
Average	5.4293	7.5689	7.5691	5.6990
SD	0.2616	0.000303	0.0448	0.2102

a better performance in decreasing transfer time because of its robustness to uncertain network information.

Although time criteria is considered in both FDM and ASP and these two algorithms are almost the same but FDM uses ANFIS to reduce errors of uncertain context information. So FDM has a better performance in transfer time decreasing. MR uses Round Robbin selection and tasks are sent to specific cloudlets, respectively, so transfer time is the same in all time steps and doesn't differ much.

Next experiment is designed to study mean completion time of the tasks.

Mean completion time is the aggregation of transfer, waiting and executing time of the tasks in servers of cloudlets. Since FDM has a better performance in balancing the loads of the cloudlets, waiting time of the tasks in servers would be minimum. Due to the results of the previous experiment, transfer time is also minimum when using FDM; in conclusion, as it can be seen in Fig. 17 and Table 7, mean completion time of tasks using FDM would be the least compared with other algorithms. Comparing Figs. 15, 16 and 17, it can be seen that with the increase of requests in 50–100 and 300–350 periods, transfer time, completion time and drop rate get higher.

In order to study all the mentioned parameters and overall performance of the compared algorithms, a figure of merit is defined as in (24):

$$\text{FoM} = 1/(T_c + \text{DropRate} + T_t + \text{CV}_\rho) \tag{24}$$

in which all the above mentioned parameters T_c (mean completion time), *DropRate* (summation of drop rates), T_t (mean transfer time), CV_ρ (utilization CV) are normalized and considered in figure of merit. Results are shown in Fig. 18.

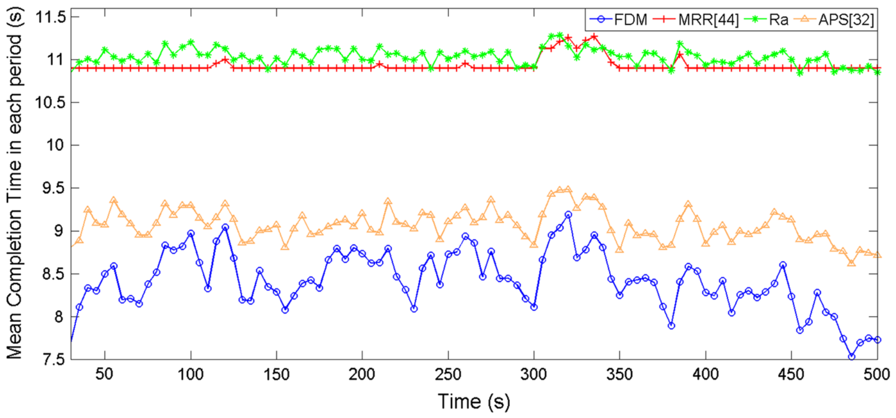


Fig. 17 Mean completion time of the tasks using different algorithms in each period

Table 7 Average and standard deviation of "Mean completion Time of the tasks" over time

	FDM	RR	Ra	ASP [12]
Average	8.4261	10.9245	10.0221	9.718
SD	0.4156	0.0865	0.0983	0.2423

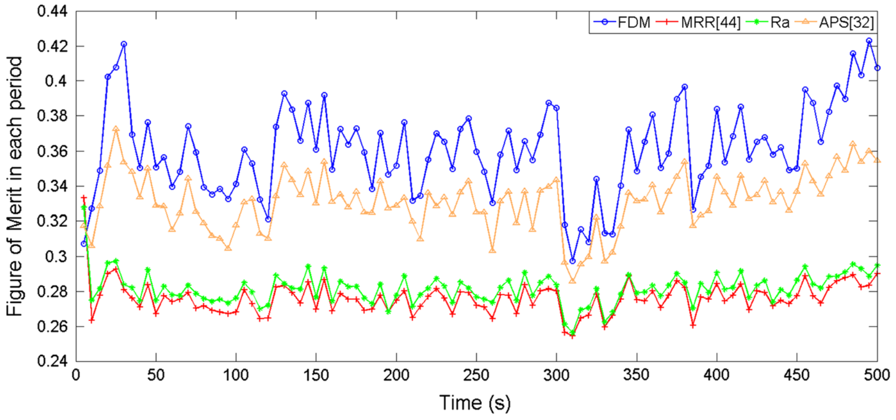


Fig. 18 Figure of merit of different algorithms in each period

Table 8 Average and standard deviation of "Figure of Merit" over time

	FDM	RR	Ra	ASP [12]
Average	0.3608	0.2761	0.2815	0.3312
SD	0.0258	0.0096	0.0091	0.0162

It is shown in Fig. 18 that FDM has a better performance over other tested algorithms because of its robustness to imprecise context information. According to the numbers inserted in Table 8, FDM excels RR, Ra, and ASP about 30, 28.17 and 8.93%, respectively.

7 Conclusion

Main goal of the proposed algorithm is calculating the probability for each host cloudlet to send a task to other cloudlets and update the host's rating table regarding obtained probabilities. These rating tables are produced considering system's goal which is load balancing of the cloudlets, drop rate reduction and task completion time reduction. The used context information should be updated in a periodic manner, so they can adapt system's new situations and adjust system behavior properly and proportionally. Since the time overhead of updating the context information needed for rating tables is high, context aware scheduling algorithms usually cannot be executed online. As a solution, context should be updated periodically, but it results in uncertainty and errors. In this paper a novel algorithm using ANFIS is proposed to reduce the errors caused by using last interval's context information for updating rating tables in the new period. Simulation results show that our proposed algorithm has a better performance over RR, Ra, and ASP [12] about 30, 28.17 and 8.93%, respectively.

References

1. Itu-t: Ict facts and figures (2014) [Online]. https://www.itu.int/en/itud/statistics/documents/facts/ictfactsfigures2014-e.pdf. Accessed 1 Sep 2014
2. Perez S (2010) Mobile cloud computing: $9.5 billion by 2014 [Online]. http://exoplanet.eu/
3. Chen M, Wu Y, Vasilakos AV (2014) Advances in mobile cloud computing. Mob Netw Appl 19(2):131–132
4. Mell P, Grance T (2011) The NIST definition of cloud computing. National Institute of Standards and Technology
5. Rahimi MR, Ren J, Liu CH, Vasilakos AV, Venkatasubramanian N (2014) Mobile cloud computing: a survey, state of art and future directions. Mob Netw Appl 19(2):133–143
6. Fernando N, Loke SW, Rahayu W (2013) Mobile cloud computing: a survey. Future Gener Comput Syst 29(1):84–106
7. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for VM-based cloudlets in mobile computing. IEEE Perv Comput 8(4):14–23
8. Dinh HT, Lee C, Niyato D, Wang P (2013) A survey of mobile cloud computing: architecture, applications, and approaches. Wirel Commun Mob Comput 13(18):1587–1611
9. Sanaei Z, Abolfazli S, Gani A, Buyya R (2013) Heterogeneity in mobile cloud computing: taxonomy and open challenges. IEEE Commun Surv Tutor 16(1):369–392
10. Shah R, Veeravalli B, Misra M (2007) On the design of adaptive and decentralized load-balancing algorithms with load estimation for computational grid environments. IEEE Trans Parallel Distrib Syst 18(12):1675–1686
11. Lu Z, Zhao J, Wu Y, Cao G (2015) Task allocation for mobile cloud computing in heterogeneous wireless networks. In: 24th International Conference on Computer Communication and Networks (ICCCN), Las Vegas, NV
12. Shi T, Yang M, Li X, Lei Q, Jiang Y (2015) An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds. Perv Mob Comput 27:90–105
13. Dai M, Liu D, Fan Y, Wang H, Lin X, Chen B, Lu Z (2016) Evolutionary study on mobile cloud computing. Neural Comput Appl 1–10 doi:10.1007/s00521-016-2217-8
14. Patel N, Chauhan S (2014) A survey on load balancing and scheduling in cloud computing. IJIRST Int J Innov Res Sci Technol 1(7):185–189
15. Chun BG, Maniatis P (2009) Augmented smartphone applications through clone cloud execution. In: 12th Conference on Hot topics in Operating Systems, Berkeley, CA
16. Park J, Yu H, Kim H, Lee E (2014) Dynamic group-based fault tolerance technique for reliable resource management in mobile cloud computing. Practice and Experience, no. special issue, Concurrency and Computation

17. Shah SC (2015) Energy efficient and robust allocation of interdependent tasks on mobile ad hoc computational grid. Concurr Comput Pract Exp 27(5):1226–1254
18. Lin Y, Chu ET, Lai Y, Huang T (2015) Time-and-energy-aware computation offloading in handheld devices to coprocessors and clouds. IEEE Syst J 9(2):393–405
19. Patel N, Chauhan S (2014) A survey on load balancing and scheduling in cloud computing. Int J Innov Res Sci Technol 1(7):185–189
20. Broch J, Maltz DA, Johnson DB, Hu Y-C, Jetcheva J (1998) A performance comparison of multi-hop wireless ad hoc network routing protocols. In New York, MobiCom ’98 Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking
21. Çevik HH, Çunkaş M (2015) Short-term load forecasting using fuzzy logic and ANFIS. Neural Comput Appl 26(6):1355–1367
22. El-Shafie A, Najah A, Karim OA (2014) Amplified wavelet-ANFIS-based model for GPS/INS integration to enhance vehicular navigation system. Neural Comput Appl 24(7–8):1905–1916
23. Cho KM, Tsai PW, Tsai CW, Yang CS (2014) A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. Neural Comput Appl 26(6):1297–1309
24. Ertunc HM, Ocak H, Aliustaog C (2013) ANN- and ANFIS-based multi-staged decision algorithm for the detection and diagnosis of bearing faults. Neural Comput Appl 22(1 Supplement):435–446
25. Flickenger R, Okay S, Pietrosemoli E, Zennaro M, Fonda C (2008) Very long distance wi-fi networks. In: Second ACM SIGCOMM Workshop on Networked Systems for Developing Regions, New York
26. Balasubramanian N, Balasubramanian A, Venkataramani A (2009) Energy consumption in mobile phones: a measurement study and implications for network applications. In: 9th ACM SIGCOMM Conference on Internet Measurement Conference, New York
27. Raiciu C, Niculescu D, Bagnulo M, Handley MJ (2011) Opportunistic mobility with multipath tcp. In: MobiArch ’11 Proceedings of the Sixth International Workshop on MobiArch, New York
28. Zadeh LA (1965) Fuzzy sets. Inf Control 8:338–353
29. Jang J-SR (1993) ANFIS: adaptive network based fuzzy inference systems. IEEE Trans Syst Man Cybern 23(3):665–685
30. Takagi T, Sugeno M (1985) Fuzzy identification of systems and its application to modeling and control. IEEE Trans Syst Man Cybern 15(1):116–132
31. Lohani AK, Goel NK, Bhatia KK (2006) Takagi–Sugeno fuzzy inference system for modeling stage-discharge relationship. J Hydrol 331(1–2):146–160
32. Arlitt MF, Williamson CL, (1996) Web server workload characterization: the search for invariants. In: Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, NY, USA
33. Baccarelli E, Vinueza Naranjo PG, Shojafar M, Scarpiniti M (2016) Q*: Energy and delay-efficient dynamic queue management in TCP/IP virtualized data centers. Comput Commun doi:10.1016/j.comcom.2016.12.010
34. Shojafar M, Cordeschi N, Abawajy JH, Baccarelli E (2015) Adaptive energy-efficient qos-aware scheduling algorithm for TCP/IP mobile cloud. In: Globecom Workshops (GC Wkshps), 2015 IEEE
35. Yang H, Luo H, Ye F, Lu S, Zhang L (2004) Security in mobile ad hoc networks: challenges and solutions. IEEE Wirel Commun 11(1):38–47