Future Generation Computer Systems 90 (2019) 149–157

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Autonomic computation offloading in mobile edge for IoT applications

Md Golam Rabiul Alam [a], Mohammad Mehedi Hassan [b,*], Md. ZIa Uddin [c], Ahmad Almogren [b], Giancarlo Fortino [d]

[a] Department of Computer Science and Engineering, BRAC University, Dhaka, Bangladesh
[b] College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
[c] Department of Informatics, University of Oslo, Norway
[d] Department of Informatics, Modeling, Electronics, and Systems, University of Calabria, 87036 Rende, Italy

HIGHLIGHTS

- An autonomic computation offloading model for mobile edge/fog is proposed.
- A deep reinforcement Q-learning model is used for computation offloading.
- Our method significantly improves the performance of the computation offloading.

ARTICLE INFO

Article history:
Received 23 February 2018
Received in revised form 13 July 2018
Accepted 25 July 2018
Available online xxxx

Keywords:
Computation offloading
Autonomic computing
Mobile edge/fog computing
Deep Q- learning

ABSTRACT

Computation offloading is a protuberant elucidation for the resource-constrained mobile devices to accomplish the process demands high computation capability. The mobile cloud is the well-known existing offloading platform, which usually far-end network solution, to leverage computation of the resource-constrained mobile devices. Because of the far-end network solution, the user devices experience higher latency or network delay, which negatively affects the real-time mobile Internet of things (IoT) applications. Therefore, this paper proposed near-end network solution of computation offloading in mobile edge/fog. The mobility, heterogeneity and geographical distribution mobile devices through several challenges in computation offloading in mobile edge/fog. However, for handling the computation resource demand from the massive mobile devices, a deep Q-learning based autonomic management framework is proposed. The distributed edge/fog network controller (FNC) scavenging the available edge/fog resources i.e. processing, memory, network to enable edge/fog computation service. The randomness in the availability of resources and numerous options for allocating those resources for offloading computation fits the problem appropriate for modeling through Markov decision process (MDP) and solution through reinforcement learning. The proposed model is simulated through MATLAB considering oscillated resource demands and mobility of end user devices. The proposed autonomic deep Q-learning based method significantly improves the performance of the computation offloading through minimizing the latency of service computing. The total power consumption due to different offloading decisions is also studied for comparative study purpose which shows the proposed approach as energy efficient with respect to the state-of-the-art computation offloading solutions.

1. Introduction

The massive growth of mobile devices (e.g. smart phones, laptops, tablet pc's, mobile IoT's and automobiles) and their computation demands imposed a huge scarcity in communication network and computation resources. Some of the application services e.g. image processing and real-time translation services require extensive computation, the resource-constrained mobile devices are not the feasible domiciles to process those applications. Therefore, to meet the computation demands of such type of mobile devices and applications the outsourcing of computation is the demand in need.

Computation offloading is a relocation mechanism of processes or modules of software applications or systems from resource-constrained devices to the resource-rich platforms. Mobile cloud is the well-known platform for computation offloading of mobile devices. Mobile cloud computing is becoming a popular method

* Corresponding author.
E-mail address: mmhassan@ksu.edu.sa (M.M. Hassan).

https://doi.org/10.1016/j.future.2018.07.050

for mobile services e.g. mobile video games, video streaming, education, social networking, messenger and mobile healthcare services [1].

However, the key barriers to offloading computation in mobile cloud are the network bandwidth and latency. Data travels a longer hazardous path from mobile device to the mobile cloud during offloading and thus consumes huge network bandwidth [2]. The bandwidth scarcity, and internet bottlenecks and traffic congestions are the catalysts for the higher latency of offloading computation. Real-time applications are highly latency sensitive and thus it requires to compute data in a close proximity of mobile devices or users. So, mobile fog can be the effective and suitable platform for offloading mobile computation.

Fog computing [3] is introduced by Cisco Systems Inc. to extend the cloud computing paradigm to the edge of network especially for Internet of Things (IoT) services. Mobile Fog is the complementary model of fog computing especially prototyped for seamless and latency-aware mobile services [4]. However, the key research questions for offloading computation in mobile fog are (1) How to offload computation in the mobile fog? (2) Which module or process of mobile application should offload? (3) Where to offload the module or process for minimizing the latency of service computing? Moreover, the mobility, heterogeneity and geographical distribution mobile devices impose additional challenges of computation offloading in mobile fog. This research contributes to finding the answer to the above questions. The key contributions of this research are as follows.

- A code offloading framework is proposed for computation offloading in mobile fog environment. The code analyzer unit of the framework determines which basic blocks of the code are computation hungry and subject to offload.
- A deep Q-learning [5] based computation offloading method is proposed for the autonomic management of massive offloading request. The trained code offloader unit of the proposed framework takes the offloading decision considering resource demand, resource availability and network status to minimize the latency of service computing.
- The performance of the proposed model studied through simulation. The performance gain in terms of latency and energy efficiency justifies the dominance of the proposed autonomic offloading model.

Rest of the paper is organized as follows. In Section 2, we discussed the related works. The system model of mobile fog is presented in Section 3. The deep Q-learning based autonomic code offloading method in mobile fog is illustrated in Section 4. We presented the simulation and performance study results in Section 5. Finally, we concluded the paper in Section 6 with some future directions.

2. State-of-the-arts computation offloading methods

Mobile fog interplays with tradition cloud to access its huge computational resources. Thus, this section discussed state-of-the-arts resource provisioning methods of legacy cloud computing paradigm. Afterwards, the pioneer works on computation offloading in the mobile cloud are discussed in this section.

The elasticity and scalability of cloud computing are achieved through virtualization of cloud resources. The resources of cloud data centers are managed through VM configuration and placement methods. The optimized placement of virtual machines in cloud brokering architecture is proposed in [6]. The paper presented very detail architecture of cloud service broker. The optimized selection of virtual resources of cloud brokers through cloud scheduler was one of the primary objectives of the paper. The holistic approach, OPTIMIS, is proposed in [7] to optimize the service lifecycle of cloud service provisioning. The paper introduced a toolkit for reliable, sustainable and trustful service provisioning.

The cost-effective deployment of computing clusters in multi-cloud infrastructure is presented in [8]. They provided analysis on the viewpoint of performance and cost. The proposal is only for loosely coupled many-task computing (MTC) applications. The proposal overlooks tightly coupled MTC applications, where facts are highly interdependent and synchronization among the computational units is necessary.

The optimal allocation of computing and networking resources in cloud computing networks is proposed in [9]. The authors of this paper used mixed integer programming to formulate optimal networked cloud mapping problem. In the proposal, the authors modeled cloud request as undirected graph of virtual nodes and virtual network links and then allocate QoS-aware virtual resources according to networked cloud request.

Energy-aware resource allocation and provisioning methods are discussed in [10]. They proposed a green cloud architecture with power model, VMs placement and migration algorithm. The proposal is fully devoted to power-aware policy development by minimizing the migration of VMs among multi-cloud infrastructure.

A joint or coordinated VM resource provisioning and maintenance scheduling method is proposed by the authors of [11]. They formulated the problem as an Integer Linear Programming problem and then transformed it into an equivalent problem to obtain linear programming relaxation solution, then they apply LIST rounding algorithm towards a final approximate solution. CoTuner [12] is the model-free reinforcement learning based VM configuration framework. It can configure VM's on the fly with changing workloads.

In mobile cloud computing, most of the pioneer works proposed the VM migration mechanism in a surrogate cloud server. The cloudlets [13] are the trusted, resource-rich and nearby computing box to offload mobile data for extensive processing. Cloudlet is well-connected to the central cloud through internet and it is considered within one hop communication range of mobile devices. The cloudlets are also called the little clouds, which act as the surrogates of centralized mobile cloud to process latency sensitive application services. The adjacent cloudlets are connected with each other through mesh connectivity [14] and can communicate and migrate virtual machines (VMs) with one other to support mobility. Each cloudlet is connected with centralized mobile cloud to fetch, store, and process necessary data through VM placement. The physical servers of host mobile cloud and cloudlets are placed in the fixed geographical locations but because of the arbitrary user requests and resource requirements the states of VMs change dynamically. The cloudlets are placed usually in coffee shops, subway stations and other public places. The dense deployment of cloudlet requires a huge investment.

CloneCloud [15] proposed a solution of offloading computation in cloud servers by introducing an automatic application partitioner, which portioned the mobile application at runtime and deploys it onto device clones in the computational cloud. The communication latency and VM formation cause jitter in latency-sensitive applications. The mobility of mobile users is ubiquitous and thus we need more efficient solutions which ensure seamless mobile services.

MobiCloud [16] proposed the Mobile Ad Hoc Networks (MANATs) as the mobile cloud computing units, where each of the mobile nodes acts like the service node. Every service node is mirrored in virtualized cloud servers to provide secure service architecture. Scavenger [17] is the mobile cyber-foraging system to offload resource intensive jobs. The framework provides an opportunity to offload computation to nearby surrogate devices.

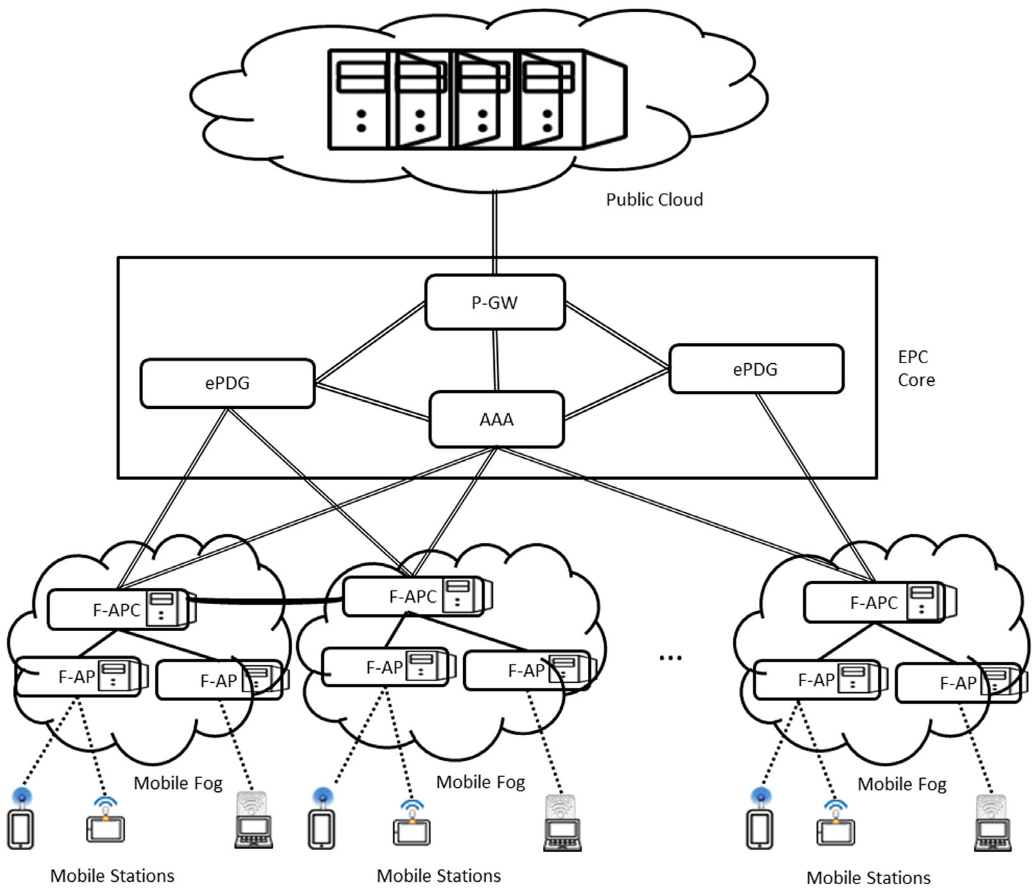


Fig. 1. The system model of mobile fog for computation offloading.

Execution offloading in mobile cloud computing is discussed in [18]. The authors' of this paper, proposed an effective way to offload useful heap objects and partial stack in the run-time of application. ThinkAir [19] proposed, a method level computation offloading mechanism for mobile cloud computing. The framework has the capability of dynamic adaptation and dynamic scaling of computational power.

In contrast to above methods, we propose deep reinforcement learning especially deep Q-learning based basic block offloading mechanism in mobile fog. The autonomic management of offloading jobs while ensuring the low latency and energy efficiency in service computing makes this proposal novel in its approach and contribution.

3. System model of mobile fog computing

Mobile Fog is the complementary model of fog computing especially prototyped for seamless and latency-aware mobile services. The system model of the mobile fog is presented in Fig. 1. The presented system model is derived from the hierarchical architecture of LTE (long-term evolution) 3GPP (3rd Generation Partnership Project) and Wi-Fi (wireless-fidelity) internetworking reference model [20].

In this architecture, the mobile fog is created on the edge of the networking modules. We consider the access point (AP) and the access point controller (APC) units as the fog nodes of mobile fog. In addition to its regular responsibilities i.e. local authentication of mobile stations, the evolved Packet Data Gateway, (ePDG) module acts as the root of the mobile fog and responsible for inter fog communication. In other words, ePDG is the collaboration unit of the mobile fog which resides in Evolved Packet Core (EPC). In this architecture, AP is not only responsible for providing connections between mobile stations (STA) and IP networks but also having sufficient storage, processing, I/O and networking capability to provide mobile cloud services e.g. IaaS, PaaS, and NaaS etc. We consider IEEE 802.11 WLAN interface between STA and AP, and IEEE Ethernet interface between AP and APC.

Similar to the AP, the APC is not only responsible for communication handovers but also responsible for code block migration to support stations mobility in mobile cloud and having sufficient storage, processing, I/O and networking capability to provide mobile cloud services as well. Therefore, the APC are also considered as the fog network controller (FNC). The upward and downward entities are interfaced with IEEE Ethernet interfacing standards. In Fig. 1, the fog enabled AP and APC are symbolized as F-AP and F-APC. The 3GPP AAA (3rd generation partnership project's authentication, authorization and accounting server) is responsible for global authentication of mobile stations of a mobile fog through EAP—AKA (Extensible authentication protocol—authentication and key agreement) over IKEv2 (Internet key exchange protocol version 2) as obtaining authentication vector from home subscriber server (HSS) unit of LTE network. We assume the Diameter as the AAA protocol in our mobile IP-based networks [21]. The P-GW (Packet data network-gateway) enables packet data network (PDN) access for user equipment's (UE)or STAs and also responsible for inter ePDG virtual machine (VM) migration in mobile fog computing. Public cloud is the traditional service delivery network of scalable, ubiquitous and pay-as-you-go services which can be accessed through the internet from static and mobile devices. If necessary, the mobile fog can utilize the required everything as a service (XaaS) of public cloud e.g. to leverage computational loads, to process latency-insensitive data, to archive transactional history

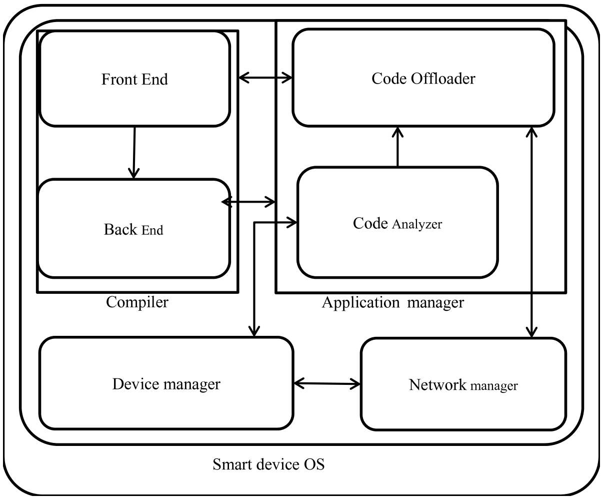


Fig. 2. The code offloading framework for offloading computation.

etc. The mobile fog can also interplay with public cloud to deliver cost-effective, ubiquitous and scalable mobile services.

4. Deep Q-learning based autonomic computation offloading

This section discusses the approach of offloading computation in mobile fog based on the system model presented in Section 3. The basic block [22] migration policy is used through mobile agents for offloading code from resource-constrained mobile stations to resource richer mobile fog. The programming code is partitioned into code generation and optimization unit of F-APC and deployed in different F-AP and also in different F-APC. According to the flow graph [22–24] of the generated codes, the basic blocks are executed on various fog nodes, where independent basic blocks are executed in a parallel fashion. To balance loads of different fog nodes, the basic blocks are migrated in different nodes within the same fog or in neighboring (or distant) fog. The communication among the mobile fogs is controlled by the ePDG node but offloading and migration performed through the tunnel between the F-APC nodes to support mobility of the mobile stations. The basic blocks can be migrated through ePDG node in a distant mobile fog for load balancing and load sharing. The basic blocks are synthesized together in case of necessary docker-container deployment in public cloud.

4.1. Code offloading framework

The traditional mobile devices have no built-in framework for offloading computation. Therefore, a middle-ware is required on top of the smart-devices operating system to perform code offloading. The proposed code offloading framework is presented in Fig. 2, where the compiler translates the high level language of applications to machine understandable form. The front end performs syntax and semantic analysis, and also generates intermediate codes. The back end of the compiler generates byte code, groups the independent byte code syntax to form basic blocks and prepares the flow graph from the basic blocks as shown in Fig. 3.

It is assumed that for the first time the application executes on the host smart device to collect the run time statistics through execution analyzer module. The execution analyzer module prepares a table of usage resources and execution time of each basic block as shown in Table 1. The application manager assigns Application ID, Method ID and Block ID of each basic block. The execution manager also keeps the record of average memory usage, CPU utilization and execution duration and also number of times the basic block

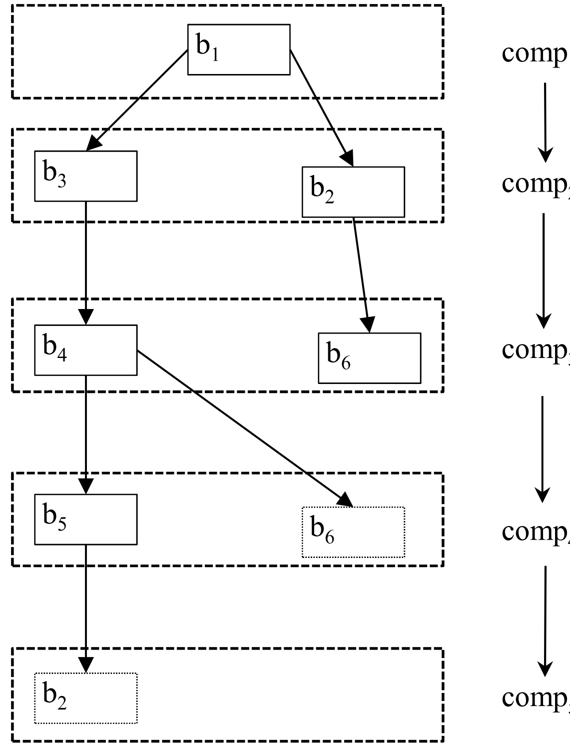


Fig. 3. The flow-graph of an example problem (i.e. the simulated example for performance analysis) with 6 basic blocks, where register values of block b1 is used by block b3 and b4. Then register values of block b2 is used by block b6 and so on. The blocks are grouped into 5 compatible sets i.e. those blocks are completely independent and feasible for parallel execution.

executes in a single run. Moreover, it also defines a basic block as not offloadable (i.e. set the offloadable flag 0) if the block requires dedicated peripherals from the smart devices (e.g. smart devices camera) during its execution.

The code offloader module is responsible for offloading the codes to nearby Fog nodes. The availability of Fog nodes are realized through network manager. It only offloads the offloadable basic blocks if necessary. If the average CPU and memory utilization of the basic block is less than the available memory and CPU then application manager executes that basic block on the host smart device. Otherwise it determines the expected execution time of host processing (i.e. EET_H) based on the available memory and CPU of the host smart device. Then it also requests F-APC for the expected execution time of Fog processing (i.e. EET_F) based on the available memory and CPU and link bandwidths of the mobile Fog by sending the history of CPU and memory usage of hosts while the block processed initially in the host. Then the code offloader compare these two expected execution time results i.e. EET_H and EET_F and make the offloading decision if $EET_F < EET_H$. Additionally, code offloader performs Breadth First Search (BFS) on the flow graph of back end compiler and find out the independent blocks of same depth and offloaded in mobile Fog for parallel execution.

To support the mobility of both mobile stations and Fog nodes, the basic blocks are migrated in different nodes within the same Fog or in neighboring (or distant) Fog. The communication between two mobile Fogs is controlled by the ePDG node but offloading and migration performed through the tunnel between the F-APC nodes to support mobility of the mobile stations. The basic blocks can be migrated through ePDG node in a distant mobile Fog also for load balancing and load sharing purpose.

4.2. Markov decision process for deep Q-learning model

As presented in the system model of mobile fog in Fig. 1, the mobile fogs are geographically distributed. The distributed fog network controller (FNC) or F-APC scavenging the available fog resources i.e. processing, memory, network to enable fog computation service. The randomness in the availability of resources and numerous options for allocating those resources for offloading computation fits the problem appropriate for modeling through Markov decision process (MDP) and solution through reinforcement learning.

According to the system model, three different sites are considered as feasible platform for offloading computation (1) the mobile fog in close proximity of end user devices, i.e. site L_1 (2) the adjacent mobile Fog (or distant mobile Fog) to handle mobility and load balancing issues, i.e. site L_2 (3) the remote public cloud to manage huge traffic and computing requirements and archiving, i.e. site L_3.

Intuitable, the deep Q-learning agent will find the best suitable place for offloading computation among the three feasible sites. Therefore, the possible action space A of the learning agent can be defined as (1) a_1: offload in location L_1 (2) a_2: offload in location L_2 (3) a_3: offload in location L_3 (4) a_4: migration from L_1 to L_2 (5) a_5: migration from L_2 to L_1 (6) a_6: migration from L_1 to L_3 (7) a_7: migration from L_3 to L_1 (8) a_8: migration from L_2 to L_3 (9) a_9: migration from L_3 to L_2 (10) a_{10}: migration within L_1. So, there are total 10 possible actions the learning agent can perform.

Now, the state space S for the learning agent can be defined as the vector of memory, processing, and networking bandwidth capability of the mobile fog. The fog network controller (FNC) or F-APC holds the learning engine and scavenging the available fog resources in a defined time slot. So, the state space can be represented as $S = s_1, s_2, \ldots, s_n$ where $s_i = (mem_i, cpu_i, band_i), i = 1 \ldots n$.

In the considered environment, a particular learning agent does not have the knowledge of global state space i.e. state spaces of all other mobile Fogs; the agent only has the knowledge of its local state space. The agents can communicate and collaborate with each other to offload basic blocks in a best suitable mobile fog or int the public cloud.

4.3. Reward function and deep Q-learning based computation offloading

The primary goal of the computation offloading is minimizing latency of processing each of the basic blocks, which mainly depends on the available processing and memory capability of a mobile fog node and the communication bandwidth. While the offloading request placed to the fog network controller (FNC), it should be on the request queue of the mobile fog. The FNC determines its processing capability by observing its state space and estimate the expected response time through queueing theoretic analysis.

To determine the estimated response time to process the basic block, the $M/M/1/K$ queueing model is considered. According to the queueing model each of the fog nodes has a single server, the maximum number of blocks it can process is K including one under service, the arrival rate ϕ of processing request follows the Poisson distribution, and the service time μ follows the Exponential distribution i.e. inter-arrival and service time has memoryless property. Therefore, according to the queueing theory [3] the expected response time $E[T]$ of a fog node in L_1 is shown in (1).

$$E[T]_{L_1} = \frac{E[N]}{\phi\,(1 - P_K)} \tag{1}$$

Where, expected number of blocks on the fog node $E[N]$ and steady-state distribution or stationary probability of finding K blocks on the queue P_K, and the probability of busy fog node ρ, and the probability of zero basic blocks on the queue P_0 can be defined as in the following Eqs. (2), (3), (4), and (5).

$$E[N] = \frac{\left(\frac{\phi}{\mu}\right)\left(1 - (K+1)\left(\frac{\phi}{\mu}\right)^{K} + K\left(\frac{\phi}{\mu}\right)^{K+1}\right)}{\left(1 - \left(\frac{\phi}{\mu}\right)\right)\left(1 - \left(\frac{\phi}{\mu}\right)^{K+1}\right)} \tag{2}$$

$$P_k = \frac{\rho^k}{\sum_{i=0}^{K} \rho^i},\ k = 0, \ldots, K \tag{3}$$

$$P_0 = 1 - \rho \tag{4}$$

$$\rho = \frac{\phi}{\mu} \tag{5}$$

If the learning agent considers to deploy a basic block in location L_2, that is in adjacent or remote mobile fog then the considered queueing model is $M/M/c/K$, where c is the number of servers in other fogs. The expected response time $E[T]$ of a fog node in L_2 can be determined through (6).

$$E[T]_{L_2} = \frac{E[u] + \rho(1 - P_K)}{\phi(1 - P_K)} + l_{1,2} \tag{6}$$

Where, expected queue length $E[u]$ of a mobile fog in L_2, and steady-state distribution or stationary probability of finding K blocks on the queue P_K, and the probability of busy fog node ρ, and the probability of zero basic blocks on the queue P_0 can be defined as in the following Eqs. (9), (7), (5), and (8). Here, $l_{1,2}$ is the communication latency between L_1 and L_2.

$$P_K = \frac{\phi^n}{c^{n-c} c! \mu^n} P_0 \tag{7}$$

$$P_0 = \begin{cases} \left[\dfrac{\rho^c \left(1 - \left(\frac{\rho}{c}\right)^{K-c+1}\right)}{c!\left(1 - \left(\frac{\rho}{c}\right)\right)} + \displaystyle\sum_{n=0}^{c-1} \frac{\rho^n}{n!}\right]^{-1}, & \mathit{if}\ \left(\dfrac{\rho}{c}\right) \neq 1 \\ \left[\dfrac{\rho^c}{c!}(K - c + 1) + \displaystyle\sum_{n=0}^{c-1} \frac{\rho^n}{n!}\right]^{-1}, & \mathit{if}\ \left(\dfrac{\rho}{c}\right) = 1 \end{cases} \tag{8}$$

$$E[u] = \frac{P_0}{c!\left(1 - \left(\frac{\rho}{c}\right)\right)^2}\left[1 - \left(\frac{\rho}{c}\right)^{K-c+1} - \left(1 - \left(\frac{\rho}{c}\right)(K - c + 1)\left(\frac{\rho}{c}\right)^{K-c}\right)\right] \tag{9}$$

If the learning agent considers to deploy a basic block in location L_3, that is in public cloud, the considered queueing model is $M/M/c/\infty$, where c is the number of servers in public cloud, and unlimited buffer size. The expected response time $E[T]$ of a cloud node in L_3 can be determined through (10).

$$E[T]_{L_3} = \frac{1}{\mu} + \frac{1}{\mu\,(c - \rho)} \sum_{i=n}^{\infty} P_0 \frac{\rho^i}{c!\left(c^{k-c}\right)} + l_{1,3} \tag{10}$$

Where, the probability of zero basic blocks on the queue is P_0, and the utilization factor ρ, can be defined as in the following Eqs. (5) and (11). Here, $l_{1,3}$ is the communication latency between L_1 and

Table 1
History of resource usages per basic block.

Application ID	Method ID	Block ID	Memory (K)	CPU (%)	Execution Duration (ms)	Number of Execution (times)	Offloadable
A0001	M001	b0001	2790	07	152	5	1
A0001	M001	b0002	1564	05	143	4	1
A0001	M001	b0003	253	01	14	25	1
A0001	M001	b0004	10,342	34	918	24	1
A0001	M000	b0001	276	01	19	1	0
A0001	M000	b0002	418	02	72	1	0

L_3.

$$P_0 = \left(\sum_{i=0}^{c-1} \frac{\rho^l}{l!} + \frac{\rho^c}{c!\left(1-\frac{\rho}{c}\right)} \right)^{-1} \tag{11}$$

Thus, the estimated response time $E_{i,j}$ can be determined through (12).

$$E_{i,j} = \begin{cases} \dfrac{E[N]}{\phi\,(1-P_K)}; \\ \quad If \;\; i = \{1, 5, 7, 10\} \;\; where \; a_i \in A \; and \; s_j \in S \\ \dfrac{E[u] + \rho(1-P_K)}{\phi(1-P_K)} + l_{1,2}; \\ \quad If \;\; i = \{2, 4, 9\} \;\; where \; a_i \in A \; and \; s_j \in S \\ \dfrac{1}{\mu} + \dfrac{1}{\mu\,(c-\rho)} \displaystyle\sum_{i=n}^{\infty} P_0 \frac{\rho^i}{c!\left(c^{k-c}\right)} + l_{1,3}; \\ \quad If \;\; i = \{3, 6, 8\} \;\; where \; a_i \in A \; and \; s_j \in S \end{cases} \tag{12}$$

While determining the response time of offloading computation in L_1, L_2 or L_3. The learning agent should learn where to offload for quicker response time. For every best placement i.e. offloading the agent is rewarded with $R(s_j, a_i)$ as in (13).

$$R(s_j, a_i) = \frac{E_{SLA}}{E_{i,j}} - P_{SLA} \tag{13}$$

$$P_{SLA} = \begin{cases} \dfrac{E_{i,j}}{E_{SLA}}; & If \; E_{i,j} > E_{SLA} \\ 0\,; & Otherwise \end{cases} \tag{14}$$

The learning agent will receive punishment P_{SLA} for the violation of service level agreement of response time as in (14), where E_{SLA} represents the threshold of response time as service level agreement.

To train the learning agent, the Q-learning approach is used, which is in the family of reinforcement learning. Therefore, the agent tries to explore the environment (here, the state space, S) and perform different actions from the action space A and observing the reward. As per the characteristics of reinforcement learning, in respect to the state space, the agent tries to perform the similar action if the agent receives reward and try to avoid those actions which causes it to pay penalty. The Q-learning worked as state–action pairs $Q(s_j, a_i)$ and it learns optimal policy without knowing the internal probabilistic model. Then based on the learning i.e based on the Q-table it can perform best action to the environment by using optimal policy. So, finding the optimal policy is the goal of the agent. However, optimal policy derived from Q-values, and therefore approximating Q-value is the key function of policy definition.

As the state space of computation offloading is vast. The possible combination of memory, CPU and bandwidth configuration are huge especially considering the fractional quantities. Therefore, the deep Q-learning model is applied to approximate Q-values and minimize the temporal difference (T_d) in (15).

Table 2
Simulation parameters of deep Q-learning based computation offloading.

Sites	Parameters	Values
Mobile stations	Total number	30
	Memory per node	1 GB
	Processor	900 MHz
	Bandwidth	100 Mbps
Fog nodes	Total Number	8
	Memory per node	16 GB
	Processor	1.6 GHz
	Bandwidth	1 Gbps
	Average hops	2
Cloud nodes	Total Number	1 of 8 VMs
	Memory per VM	64 GB
	Processor	2.44 GHz
	Bandwidth	10 Gbps
	Average hops	13

$$T_d(a_t, s_t) = R(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{15}$$

Where, $R(s_t, a_t)$ is the reward for current action can be determined through (13); $Q(s_t, a_t)$ is the current Q-value and $Q(s_{t+1}, a_{t+1})$ is the future Q-values with discount factor γ. The applied deep Q-learning model is presented in Fig. 4.

5. Performance evaluation

The performance of proposed computation offloading in mobile Fog is evaluated through simulation study. In the simulation topology, two adjacent mobile Fogs are connected and each of mobile fog contains one FNC or F-APC and three F-AP Fog nodes. Both of the Fog nodes are connected to the cloud node with eight VMs. We mostly focus on two performance criteria: response time and energy consumption. The simulation parameters are presented in Table 2. The flow graph of our studied benchmark application i.e. N-queen problem is presented in Fig. 3. The resource usage history of the N-queen problem is also presented in Table 1, where the value of N is 4.

Fig. 5 shows that smart phone takes longer time to process the benchmark application, whereas the mobile fog takes shorter time to place the Queens on the board. The cloud also takes less time to process the solution because of the execution power of cloud servers. It shows that computation offloading is sutiable for faster processing. Fig. 6 shows the energy consumption break downs of different computing environment. Without offloading data the processor, display unit and other peripherals of smart phone consumes huge energy. That is, they consume much energy which may degrade the mobiles battery life. In contrast, cloud and fog can compute without display and with low power consumption unit. Fog consumes lowest energy because of the closest proximity of fog nodes reduce the radio energy consumption.

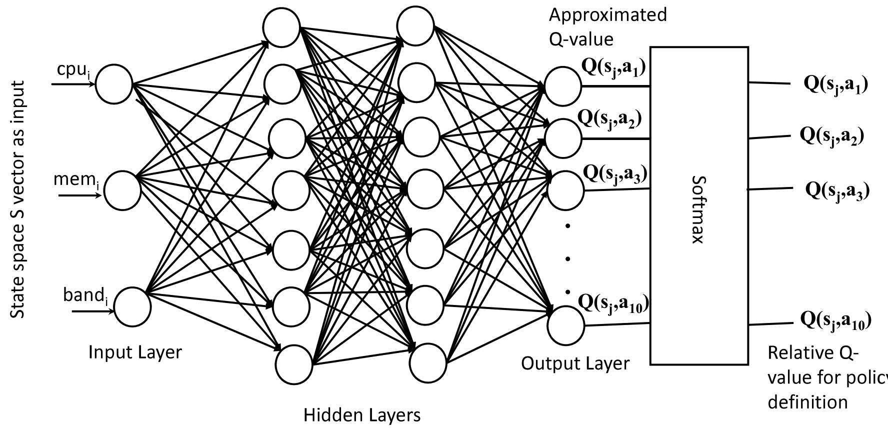


Fig. 4. The deep Q-learning architecture for approximating Q updates.

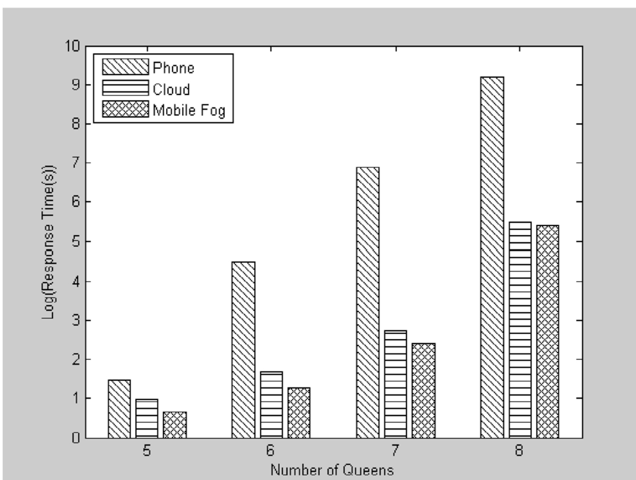


Fig. 5. The log-normal response times of benchmark N-queen problem. We present the simulation results with 5 to 8 numbers of queens. Beyond 8 queens are not suitable to execute on smart phone because of longer execution time and up to 4 queens puzzle are not suitable for computation offloading because of low computational load.

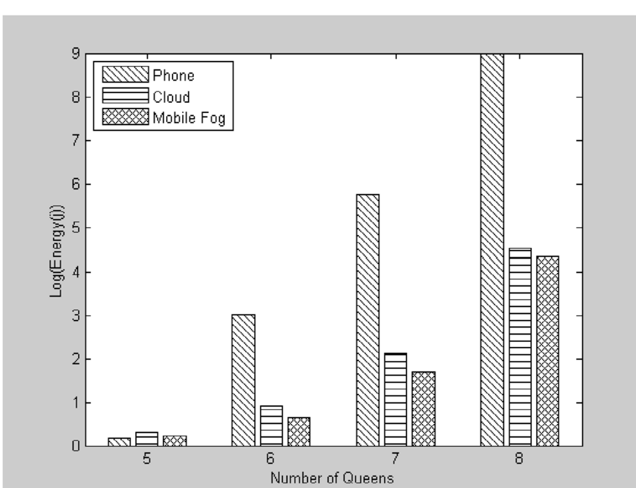


Fig. 6. The log-normal energy consumptions of different computing models to generate outputs for different number of Queens of N-Queen puzzle.

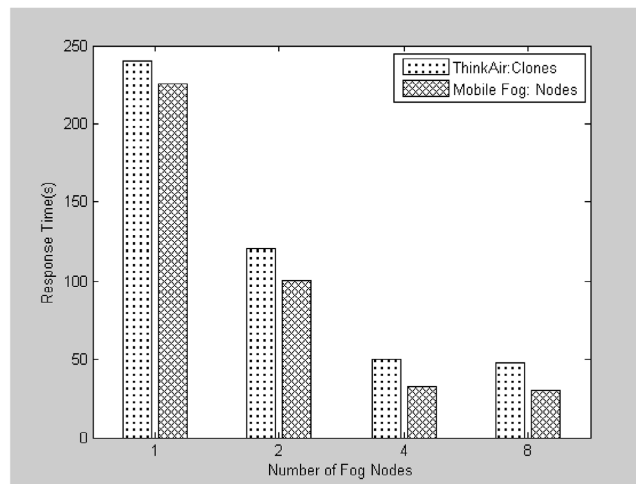


Fig. 7. Comparative study with existing benchmark solution of mobile cloud computing. ThinkAir deployed up to 8 colons to solve the 8-Queen problem, whereas Mobile Fog deployed 8 fog nodes to solve 8-Queen puzzle.

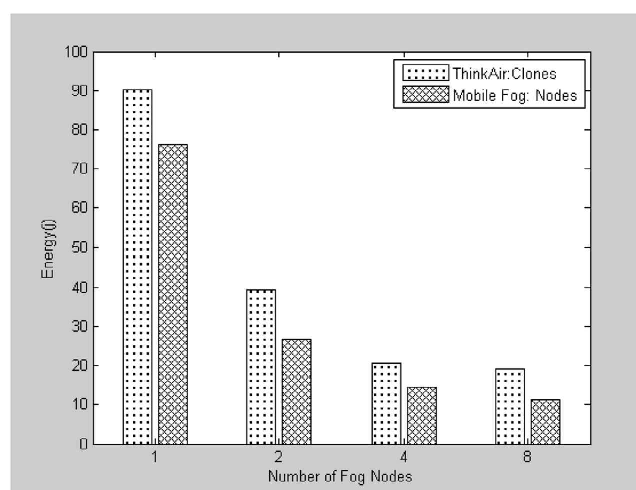


Fig. 8. Parallel execution of offloaded computation reduces the energy consumption both in ThinkAir and Mobile Fog computing.

The response time of remote cloud is always higher than the response time of mobile fog because mobile fog is nearer to the mobile stations and remote cloud is generally far from the mobile devices. Another important aspect of our proposed basic block offloading mechanism is the ability of parallel execution. ThinkAir

[19] also executed different methods concurrently, but our offloading mechanism can execute the insider basic blocks of a method in a parallel fashion. Thus basic blocks have more parallelism options, which leads lower response time and lower energy consumption as shown in Figs. 7 and 8.

6. Conclusion

The proposed deep Q-learning based code offloading method leverage the mobile cloud computing. As it is a multi-agent based distributed method, agents learn from the environment through reinforcements. The offloading method deploys basic blocks in compatible fog nodes to support parallelism. The experimental results show the improved performance of the proposed offloading method in respect to execution time and latency and energy consumption.

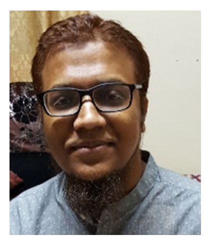
Acknowledgment

The authors extend their appreciation to the Deanship of Scientific Research, King Saud University, Saudi Arabia for funding this work through research group no (RGP- 1437-35).

References

[1] N. Fernando, S.W. Loke, W. Rahayu, Mobile cloud computing: a survey, Future Gener. Comput. Syst. 29 (1) (2013) 84–106.
[2] H.T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, Wirel. Commun. Mob. Comput. 13 (18) (2013) 1587–1611.
[3] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of The First Edition of the MCC Workshop on Mobile Cloud Computing, ACM, 2012, pp. 13–16.
[4] M.G.R. Alam, Y.K. Tun, C.S. Hong, Multi-agent and reinforcement learning based code offloading in mobile fog, in: Information Networking, ICOIN, 2016 International Conference on, IEEE, 2016, pp. 285–290.
[5] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[6] J. Tordsson, R.S. Montero, R. Moreno-Vozmediano, I.M. Llorente, Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers, Future Gener. Comput. Syst. 28 (2) (2012) 358–367.
[7] A.J. Ferrer, F. HernáNdez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R.M. Badia, K. Djemame, et al., Optimis: a holistic approach to cloud service provisioning, Future Gener. Comput. Syst. 28 (1) (2012) 66–77.
[8] R. Moreno-Vozmediano, R.S. Montero, I.M. Llorente, Multicloud deployment of computing clusters for loosely coupled mtc applications, IEEE Trans. Parallel Distrib. Syst. 22 (6) (2011) 924–930.
[9] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, A. Monje, On the optimal allocation of virtual resources in cloud computing networks, IEEE Trans. Comput. 62 (6) (2013) 1060–1071.
[10] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. 28 (5) (2012) 755–768.
[11] Z. Zheng, M. Li, X. Xiao, J. Wang, Coordinated Resource Provisioning and Maintenance Scheduling in Cloud Data Centers, IEEE, 2013.
[12] X. Bu, J. Rao, C.-Z. Xu, Coordinated self-configuration of virtual machines and appliances using a model-free learning approach, IEEE Trans. Parallel Distrib. Syst. 24 (4) (2013) 681–690.
[13] R.-P.M. Hardware, The case for vm-based cloudlets in mobile computing.
[14] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, K. Ha, The role of cloudlets in hostile environments, IEEE Pervasive Comput. 12 (4) (2013) 40–49.
[15] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, Clonecloud: elastic execution between mobile device and cloud, in: Proceedings of the Sixth Conference on Computer Systems, ACM, 2011, pp. 301–314.
[16] D. Huang, X. Zhang, M. Kang, J. Luo, Mobicloud: building secure cloud framework for mobile computing and communication, in: Service Oriented System Engineering, SOSE, 2010 Fifth IEEE International Symposium on, Ieee, 2010, pp. 27–34.
[17] M.D. Kristensen, Scavenger: transparent development of efficient cyber foraging applications, in: Pervasive Computing and Communications, PerCom, 2010 IEEE International Conference on, IEEE, 2010, pp. 217–226.
[18] S. Yang, D. Kwon, H. Yi, Y. Cho, Y. Kwon, Y. Paek, Techniques to minimize state transfer costs for dynamic execution offloading in mobile cloud computing, IEEE Trans. Mob. Comput. 13 (11) (2014) 2648–2660.
[19] S. Kosta, A. Aucinas, P. Hui, R. Mortier, X. Zhang, Thinkair: dynamic resource allocation and parallel execution in the cloud for mobile code offloading, in: Infocom, 2012 Proceedings IEEE, IEEE, 2012, pp. 945–953.
[20] C. Yoo, Network architecture for lte and wi-fi internetworking, NMC consulting group..
[21] N. Kottapalli, Diameter and lte evolved packet system, Radisys, White Paper [Online] go.radisys.com. (Accessed: 5 September 2015).
[22] A.V. Aho, R. Sethi, J.D. Ullman, Compilers: Principles, Techniques, and Tools, vol. 2, Addison-wesley, Reading, 2007.
[23] X. Li, D. Li, J. Wan, C. Liu, M. Imran, Adaptive transmission optimization in sdn-based industrial internet of things with edge computing, IEEE Internet Things J. (2018).
[24] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, M. Qiu, A scalable and quick-response software defined vehicular network assisted by mobile edge computing, IEEE Commun. Mag. 55 (7) (2017) 94–100.

Md Golam Rabiul Alam received B.S. and M.S. degrees in Computer Science and Engineering, and Information Technology respectively. He also received Ph.D. in Computer Engineering from Kyung Hee University, Korea in 2017. He is currently working as a Post-doctoral researcher in Computer Science and Engineering Department at Kyung Hee University, Korea. His research interest includes healthcare informatics, mobile cloud computing, ambient intelligence and persuasive technology.



Mohammad Mehedi Hassan is currently an Associate Professor of Information Systems Department in the College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Kingdom of Saudi Arabia. He received his Ph.D. degree in Computer Engineering from Kyung Hee University, South Korea in February 2011. He received Best Paper Award from CloudComp conference at China in 2014. He also received Excellence in Research Award from CCIS, KSU in 2015 and 2016 respectively. He has published over 100+ research papers in the journals and conferences of international repute. He has served as, chair, and Technical Program Committee member in numerous international conferences/workshops like IEEE HPCC, ACM BodyNets, IEEE ICME, IEEE ScalCom, ACM Multimedia, ICA3PP, IEEE ICC, TPMC, IDCS, etc. He has also played role of the guest editor of several international ISI-indexed journals such as IEEE IoT, FGCS, etc. His research areas of interest are cloud federation, multimedia cloud, sensor-cloud, Internet of things, Big data, mobile cloud, cloud security, IPTV, sensor network, 5G network, social network, publish/subscribe system and recommender system. He is a member of IEEE.

Md. Zia Uddin received his Ph.D. in Biomedical Engineering in February of 2011. He is currently working as a postdoctoral research fellow under Dept. of Informatics, University of Oslo, Norway. Dr. Zia's researches are mainly focused on computer vision, image processing, artificial intelligence, and pattern recognition. He got more than 60 research publications including international journals, conferences and book chapters.

Ahmad Almogren has received Ph.D. degree in computer sciences from Southern Methodist University, Dallas, Texas, USA in 2002. Previously, he worked as an assistant professor of computer science and a member of the scientific council at Riyadh College of Technology. He also served as the dean of the college of computer and information sciences and the head of the council of academic accreditation at Al Yamamah University. Presently, he works as an Associate Professor and the vice dean for the development and quality at the college of computer and information sciences at King Saud University in Saudi Arabia. He has served as a guest editor for several computer journals. His research areas of interest include mobile and pervasive computing, computer security, sensor and cognitive network, and data consistency.

Giancarlo Fortino is currently a Professor of Computer Engineering (since 2006) at the Dept. of Informatics, Modeling, Electronics and Systems (DIMES) of the University of Calabria (Unical), Rende (CS), Italy. He holds the "Italian National Habilitation" for Full Professorship. He has been a visiting researcher at the International Computer Science Institute, Berkeley (CA), USA, in 1997 and 1999, and visiting professor at Queensland Univ. of Technology, Brisbane, Australia, in 2009.He was nominated Guest Professor in Computer Engineering of Wuhan Univ. of Technology (WUT) on April, 18 2012. His research interests include distributed computing, wireless sensor networks, software agents, cloud computing, Internet of Things systems. He authored over 230 publications in journals, conferences and books. He is the founding editor of the Springer Book Series on Internet of Things: Technology, Communications and Computing and serves in the editorial board of IEEE Transactions on Affective Computing, Journal of Networks and Computer Applications, Engineering Applications of Artificial Intelligence, Information Fusion, Multi Agent and GRID Systems, etc. He is co-founder and CEO of SenSysCal S.r.l., a spinoff of Unical, focused on innovative sensor-based systems for e-health and demotics. He is IEEE Senior member.