Multi-agent and Reinforcement Learning Based Code Offloading in Mobile Fog

Md. Golam Rabiul Alam, Yan Kyaw Tun, Choong Seon Hong
Department of Computer Science and Engineering
Kyung Hee University
Yongin-si, Korea
robi@khu.ac.kr, ykyawtun7@gmail.com, cshong@khu.ac.kr

***Abstract*—Fog computing, which performs on network edges, is a front-end distributed computing archetype of centralized cloud computing. Mobile Fog is a special purpose computing prototype, which leverages the mobile computing to deliver seamless and latency-aware mobile services. Offloading computation in mobile Fog is challenging because of the spatiotemporal resource requirements of heterogeneous mobile devices. In this paper, we propose reinforcement learning based code offloading mechanism to ensure low-latency service delivery towards mobile service consumers. We use the distributed reinforcement learning algorithm to offload basic blocks in a decentralized fashion to deploy mobile codes on geographically distributed mobile Fogs. We simulate the proposed prototype using OMNeT++ considering fluctuated resources of mobile Fog and varied service demands of mobile users. The proposed method significantly reduces the execution time and latency of accessing mobile services while ensuring lower energy consumption of mobile devices.**

Keywords—Fog computing; mobile computing; computation offloading; reinforcement learning

I. Introduction

Mobile cloud computing is becoming a popular method for mobile services e.g. mobile video games, video streaming, education, social networking, messenger and mobile healthcare services [1]. As mobile data traffic increases exponentially, mobile service providers utilize mobile clouds as an effective way to deliver smooth services. 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, and thus offloading computation in mobile cloud is a popular solution for smooth delivery of mobile services. However, data travels a longer hazardous path from mobile device to mobile cloud during offloading and thus consumes huge network bandwidth [2]. The cloudlets [4] are the trusted, resource-rich and nearby computing box to offload mobile data for extensive processing [17]. Cloudlet is well-connected to the central cloud through internet and it is considered within one hop communication range of mobile devices.

CloneCloud [14] proposed a solution of offloading computation in cloud servers by introducing an automatic application practitioner, which portioned the mobile application in runtime and deploy it onto device clones in computational cloud. The communication latency and virtual machine (VM) formation causes jitter in latency sensitive applications [4].

Fog computing [3][7] introduced by Cisco Systems Inc. to extend the cloud computing paradigm to the edge of the network especially for the Internet of Things (IoT) services. Mobile Fog is the complementary model of Fog computing especially prototyped for seamless and latency-aware mobile services. In this research, we propose a system model of mobile Fog and propose a distributed reinforcement learning method for offloading computation in mobile Fog.

ThinkAir [13] 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 of above methods, we propose reinforcement learning based basic block offloading mechanism in mobile Fog. The proposed scheme is multi-agent based distributed method for offloading computation in dispersed mobile Fogs.

Offloading computation in mobile Fog is challenging because of the dynamicity of mobile environment i.e. user traffics, mobility, network condition, and SLAs of different services. We consider Markov Decision Process (MDP) to model such a dynamic environment appropriately. In such a pulsating discrete event system model the transition probabilities among different states of Fog nodes are unknown, thus we formulate and solve the offloading problem using distributed reinforcement learning [5][15] approach.

II. System Model of Mobile Fog

Mobile cloud computing is a data and process offloading platform of mobile stations to ensure quality services toward the mobile users. In Fig. 1, we introduce a model of mobile cloud computing based on the Fog computing [7] model of Cisco Systems, Inc. In this mobile Fog computing model we use the hierarchical architecture of LTE (long term evolution) and Wi-Fi (wireless-fidelity) internetworking reference model [8].

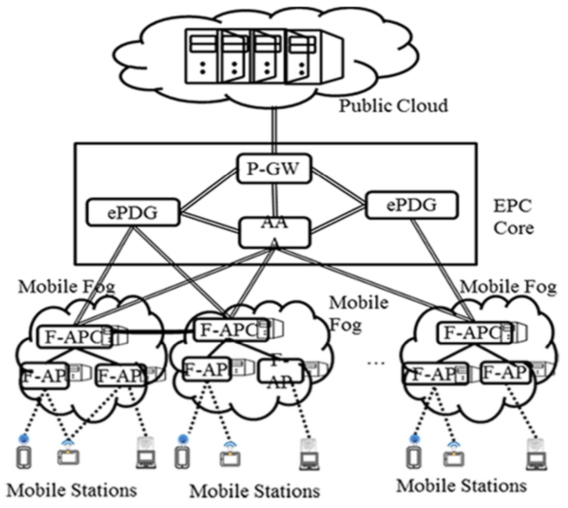


Fig. 1. Mobile Fog based mobile cloud computing service delivery architecture.

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. 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 [8]. 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 internet from static and mobile devices [14]. 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 etc. The mobile Fog can also interplay with public cloud to deliver cost effective, ubiquitous and scalable mobile services.

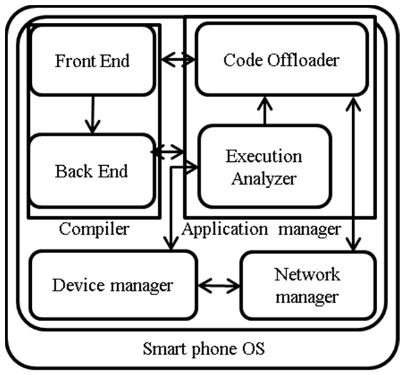


Fig. 2. The code offloading framework of Smart phone.

III. Reinforcement Learning based Code Offloading in Mobile Fog

We discussed the architecture of mobile Fog in section II, in this section we discuss the approach of offloading computation in mobile Fog. We use basic block [11] migration policy through mobile agents for offloading code from resource constrained mobile stations to resource richer mobile Fog.

A. *Code offloading framework of Smart devices*

The code offloading framework of smart devices 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.

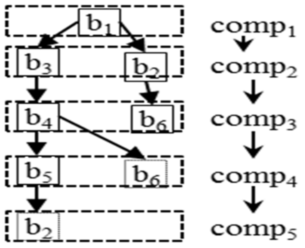


Fig. 3. The flow-graph of the N-queen problem (i.e. the implemented example for performance analysis) with 6 basic blocks, where register values of block b_1 is used by block b_3 and b_4. Then register values of block b_2 is used by block b_6 and so on. The blocks are grouped into 5 compatible sets i.e. those blocks are completely independent and feasible for parallel execution. .

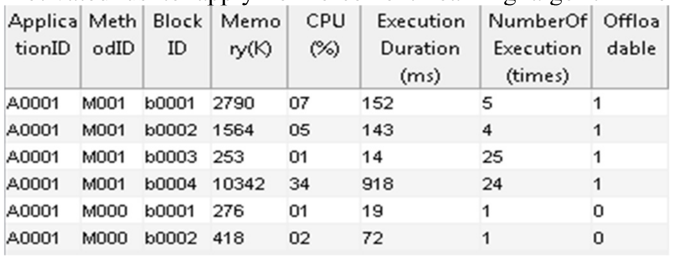
We assume that for the first time the application executes on the host smart device to collect the run time statistics through execution analyzer module. The execution analyzer module prepares a table of usage resources and execution time of each basic block as shown in Fig. 4. The application manager 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 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 ofloadable 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.

B. *Basic Block Offloading through Distributed Reinforcement Learning*

Mobile Fogs are geographically distributed to serve mobile stations from their close proximity. The dynamicity of resource requests from various mobile stations, applications and users motivated us to apply reinforcement learning algorithm for



ApplicationID	MethodID	Block ID	Memory(K)	CPU (%)	Execution Duration (ms)	NumberOf 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	10342	34	918	24	1
A0001	M000	b0001	276	01	19	1	0
A0001	M000	b0002	418	02	72	1	0

Fig. 4. The basic block-wise resource usage history of an application for a single run.

resource provisioning [6] of mobile Fog. The distributed nature of mobile Fog also inspired us to apply distributed reinforcement learning for mobile code offloading. We have three different options to offload codes 1) the mobile Fog in close proximity of mobile stations, L_1 2) the adjacent mobile Fog (or distant mobile Fog) to handle mobility and load balancing issues, L_2 3) the remote public cloud to manage huge traffic and computing requirements, and archiving, L_3. We deploy multiple mobile agents to find the best suitable options to offload basic blocks of mobile codes. The mobile agents will learn from environment and experience by following DFG ("Dissolution and Formation of Groups") algorithm [9][10]. The DFG algorithm is designed for action selection and learning in multi-agent system.

The mobile agents action space A in case of basic block offloading in mobile Fog is 1) a1: offload in location L_1 2) a_2: offload in location L_2 3) a_3: offload in location L_3 4) a_4: migrate from L_1 to L_2 5) a_5: migrate from L_2 to L_1 6) a_6: migrate from L_1 to L_3 7) a_7: migrate from L_3 to L_1 8i) a_8: migrate from L_2 to L_3 9) a_9: migrate from L_3 to L_2 10) a_{10}: migration within L_1. Each individual agent G_i is responsible for a specific action in our multi agent system. We group the agents to compete one another following group development and dissolving strategy of DFG algorithm [10]. A group may contain a single agent or multiple and each of the agent groups has a leader agent to manage bidding with other groups to participate in competition. For example, some agents (a group) want to deploy a basic block into the local Fog; other group wants to offload the basic block in a neighboring Fog, these two groups bidding each other with their capability of processing the basic block. The independent blocks can be executed in parallel fashion to reduce the latency of execution but the dependent blocks should be processed in sequential order. So, agent's actions set may become incompatible if concurrent execution doesn't support the flow graph of the mobile applications code.

We consider the state space S of mobile agents as the vector of storage, processing, and networking bandwidth capability of Fog nodes. 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 our considered environment, a particular agent doesn't have the knowledge of global state space i.e. state spaces of all other mobile Fogs; the agent only has the knowledge of local state space. The agents can communicate and collaborate with each other to offload basic blocks in a best suitable Fog node.

C. *Action Selection*

In section III (B), we already mentioned that an individual agent is responsible for a specific action. For example, we can consider the agents of the mobile Fog L_1 as a group and an agent is responsible for placement of basic block in F-APC, another agent is responsible for placement of basic block in F-AP_1, and another agent is responsible for placement of basic block in F-AP_2, and also the responsible agent of F-APC as the leader of this group. The group leader announces a bid $B_{i,j}$ to deploy a set of compatible basic blocks in location L_1 considering his member agents estimated capability $E_{i,j}$[9].

$$B_{i,j} = \alpha * E_{i,j} + \beta * E_{i,j} \qquad (1)$$

Here, i and j represent the agent and state respectively, α is a small constant called the risk factor of the estimation and β is a small random number called the noise factor to escape from local minima. We consider the estimated response time to process the block as the estimated capability. To determine the estimated response time to process the basic block we apply queuing theory to analyze queue status of each of the Fog nodes (i.e. F-APC and F-AP). We assume *M/M/1/K* queuing model i.e. 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 Poisson distribution, and the service time μ follows Exponential distribution i.e. inter-arrival and service time follows memoryless property. According to the queuing theory [12] the expected response time $E[T]$ of a Fog node in L_1 is:

$$E[T]_{L1} = \frac{E[N]}{\varphi(1-P_K)} \quad (2)$$

Where, E[N] is the expected number of blocks on the Fog node as in (3). The steady-state distribution or stationary probability of finding K blocks on the queue is P_K, and the probability of busy Fog node is ρ.

$$E[N] = \frac{\left(\frac{\varphi}{\mu}\right)\left(1-(K+1)\left(\frac{\varphi}{\mu}\right)^K + K\left(\frac{\varphi}{\mu}\right)^{K+1}\right)}{\left(1-\left(\varphi/\mu\right)\right)\left(1-\left(\varphi/\mu\right)^{K+1}\right)} \quad (3)$$

If the group leader considers to deploy a basic block in location L_2 , that is in adjacent or remote mobile Fog then we can model the queuing problem as *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 is:

$$E[T]_{L2} = \frac{E[Q]+\rho(1-P_K)}{\varphi(1-P_K)} + L_{1,2} \quad (4)$$

Where, expected queue length $E[Q]$ 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. Here, $L_{1,2}$ is the communication latency between L_1 and L_2. If the group leader considers to deploy a basic block in location L_3, that is in public cloud, we can model the queuing problem as *M/M/c/*α, 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 is:

$$E[T]_{L3} = \frac{1}{\mu} + \frac{1}{\mu(c-\rho)}\sum_{l=n}^{\alpha} P_0 \frac{\rho^l}{c!(c^{k-c})} + L_{1,3} \quad (5)$$

Where, the probability of zero basic blocks on the queue P_0, and the utilization factor ρ. Here, $L_{1,3}$ is the communication latency between L_1 and L_3.

Thus the estimated response time $E_{i,j}$ can be formulated as in (6), which can be considered as the initial estimation of DFG.

$$E_{i,j} = \begin{cases} \frac{E[N]}{\varphi(1-P_K)}, \\ If\ i = \{1,\ 5,\ 7,\ 10\}\ where\ a_i \in A\ and\ s_j \in S \\ \frac{E[Q]+\rho(1-P_K)}{\varphi(1-P_K)} + L_{1,2}, \\ If\ i = \{2,\ 4,\ 9\}\ where\ a_i \in A\ and\ s_j \in S \\ \frac{1}{\mu} + \frac{1}{\mu(c-\rho)}\sum_{l=n}^{\alpha} P_0 \frac{\rho^l}{c!(c^{k-c})} + L_{1,3}, \\ If\ i = \{3,\ 6,\ 8\}\ where\ a_i \in A and\ s_j \in S \end{cases} \quad (6)$$

Now, using the values of $E_{i,j}$ different agent group, say {a_1, a_5, a_7, a_{10}}, {a_2, a_4, a_9}, and {a_3, a_6, a_8} determine their bidding values according to (1). The agent group with maximum bidding value becomes the winner (active) and performs the action.

$$B_{i,j} = arg\ \max_k\{B_{k,j}\} \quad (7)$$

D. Credit Assignment

Credit assignment is the policy to pass the rewards to former winner group or agent to set up the environment. The bucket-brigade model is followed to assign credits to the predecessor. Let the agent *i* is the currently active (winner) on state space s_j and agent *k* is its immediate predecessor, which was active at state space s_l. According to bucket-brigade policy, agent *i* reduces it's $E_{i,j}$ by a fraction of the risk factor α and adds the external reward as in (8). We define the fulfillment of the service level agreement (SLA) as the external reward and SLA violation as the punishment as in (9). Finally, agent *k* receives credit from agent *i* according to (10).

$$E_{i,j} = E_{i,j} - \alpha * E_{i,j} + R \quad (8)$$

$$R = \begin{cases} \delta * \frac{E_{i,j} - RSP_{req}}{E_{i,j}},\ if\ E_{i,j} \le RSP_{req} \\ \delta * \frac{E_{i,j}}{RSP_{req}},\ if\ E_{i,j} > RSP_{req} \end{cases} \quad (9)$$

$$E_{k,l} = E_{k,l} + \alpha * E_{i,j} \quad (10)$$

Here, δ is weight factor of reward and RSP_{req} is the required response time to maintain SLA. The assigned credit, which represents the excellence of agent's action, is used in group formation as shown in *Algorithm 1.*

E. Grouping and Dissolution of Agents

In a multi-agent system, agents are responsible for different actions. Compatible action set depends on compatible agents grouping. Groups are not always fixed, based on the usefulness and performance of any agent in a group, the agent may change its group. The performance metric we consider in agent grouping is the gliding mean of estimated response time $E_{i,j}$ of last *v* episodes as in (11).

$$M_{i,j}^{[\tau+1]} = \frac{1}{v}\sum_{\epsilon=\tau-v+1}^{\tau} E_{i,j}^{[\epsilon]} \quad (11)$$

Where, ∊ is the last episode, (τ+1) is the actual episode, *i* and *j* represents the agent and state respectively.

According to the determined gliding mean we find out the agents which performance are degraded over time or which are stagnant by (12), where σ is the cooperation factor. These agents are the units called ready to cooperate to form in groups in their activity context.

$$M_{i,j}^{[\tau+1]} \le \sigma * E_{i,j}^{[\tau-v]} \quad (12)$$

The procedure of group formation and group dissolution are presented in *Algorithm 1* and *Algorithm 2* respectively. Agents compatibility is maintained in group formation because incompatible agents may performed incompatible actions. Group dissolution is also necessary to overcome agents from ill bindings or incompatible group formation. If the gliding mean of a group is below the minimum level, the group is dissolved by its group leader as shown in Algorithm 2.

We presented the algorithmic steps of basic block offloading in mobile Fog through multi-agent based distributed reinforcement learning approach, which is developed upon the basis of DFG algorithm. The appropriate sequence of actions to offload basic blocks by considering environmental states will learn by the agents through repeated execution of *Algorithm 3.*

Algorithm 1: Group_Formation(*int q*)

```
1. Determine the ready to cooperate set $U$
2. While U ≠ {∅}
3. Find $U_i$ of $E_{i,j} = \max_l\{E_{l,j}: (U_l) \in U\}$ and announce
   cooperation offer to other units.
4. Receive cooperation response from compatible $U_l$ units.
5. Among the compatible $U_l$ units, choose the maximum $q$
   units of $U_k$ as
   $E_{k,j} = \max_l\{E_{l,j}: (U_l) \in compatible\ (U)\}$
6. Define $U_i$ as the leader of the group.
7. Update $U = U \setminus \{U_i, U_k\}\ \forall_{U_k \in compatible\ (U)}$
8. End While
```

Algorithm 2: Group_Dissolution( )

```
1. For all group $G_i$
2. Determine the gliding mean according to (11)
3. If the gliding mean is below the product of dissolution
   factor $\eta$ and initial estimated response time $E_{init}$.
      $M_{i,j}^{[\tau+1]} \le \eta * E_{init}$
4.    Free the group members and consider them as in
   ready to cooperate set.
5. End for
```

Algorithm 3: DFG_Fog_Offloading( )

```
1.  Request receives mobile Fog nodes (F-AP, F-APC) to
    offload computation.
2.  Read the basic blocks and flow graphs [for processing
    requirements] of mobile code.
3.  For each set of compatible basic blocks having $q$ block,
    perform the following steps.
4.     Mobile agents of fog nodes check their actual
       environmental state (mem, cpu, band).
5.     Develop an agents group through Algorithm 1 i.e.
       Group_Formation( $q$ ).
6.     Each agent of a group estimate $E_{i,j}$ according to (6).
7.     The leaders of agent groups determine bidding value
       according to (1) and (7) to compete with one another.
8.     The group with maximum bidding value becomes
       active (winner) of this episode.
9.     The active agent performs the action accordingly.
10.    Receive credits by adjusting estimates and external
       rewards.
11.    End For
12.    Dissolve groups through Group_Dissolution( ).
```

IV. Performance Evaluation

The performance of mobile Fog is evaluated through simulation using OMNeT++. In our simulation topology, we connect two adjacent mobile Fogs each having one F-APC Fog node and three F-AP Fog nodes. Both of the Fog nodes are connected to the cloud node with eight VMs. The simulation parameters are presented in Table 1. To analyze the performance of mobile Fog we simply off some nodes and restart again when necessary. 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 Fig. 4, where the value of N is 4.

Table 1: Simulation setup with parameters and corresponding values

Nodes	Parameters	Values
Mobile stations	Total number	30
	Memory per node	1 GB
	Processor	900MHz
	Connection BW	100 Mbps
Fog nodes	Total number	8
	Memory per node	16 GB
	Processor	1.6 GHz
	Connection BW	1 Gbps
	Average hops from mobile station	2
Cloud node	Total number	1 of 8 VMs
	Memory per VM	64 GB
	Processor of Each VM	2.44 GHz
	Connection BW	10 Gbps
	Average hops from mobile station	13

We mainly studied the energy consumption and the response time (with execution time and latency time) of getting service in three different computing environment i.e. smart devices, cloud and mobile Fog.

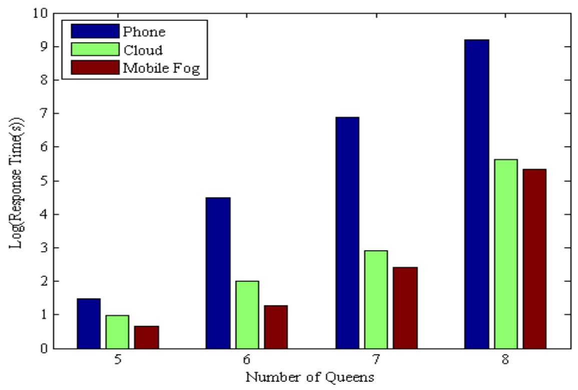


Fig. 5. The log-normal response times of benchmark N-queen problem. We present the simulation results with 5 to 8 numbers of queens. Beyond 8 queens are not suitable to execute on smart phone because of longer execution time and up to 4 queens puzzle are not suitable for computation offloading because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

Fig. 5 shows that smart devices 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. Fig. 6 shows the energy consumption break downs of different computing environment. Without offloading data the processor, display unit and other peripherals of smart devices consumes huge energy. 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. 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.

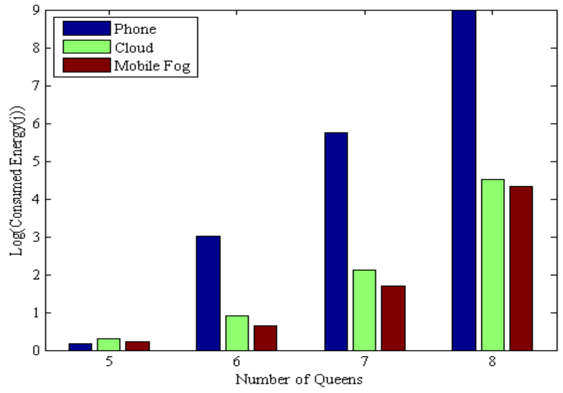


Fig. 6. The log-normal energy consumptions of different computing models to generate outputs for different number of Queens of N-Queen puzzle.

ThinkAir [13] 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 as shown in Fig. 7.

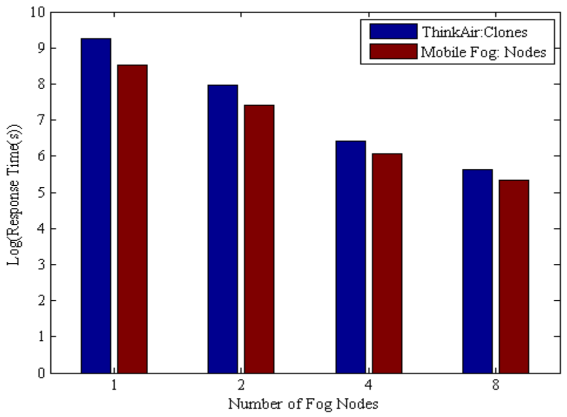


Fig. 7. Comparative study with existing benchmark solution of mobile cloud computing. ThinkAir deployed up to 8 colons to solve the 8-Queen problem, whereas Mobile Fog deployed 8 Fog nodes to solve 8-Queen puzzle.

V. Conclusion

We propose Mobile Fog architecture to leverage mobile cloud computing and then we propose a code offloading mechanism in mobile Fog. The offloading method is 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. There are some challenging issues in successful development of mobile Fog including mobility and context aware offloading. We continue our works on virtualization, mobility, privacy and end user issues of mobile Fog computing.

Acknowledgment

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2015-(H8501-15-1015)) supervised by the IITP(Institute for Information & communications Technology Promotion). *Dr. CS Hong is the corresponding author.

References

[1] N. Fernando, S. W. Loke, and W. Rahayu. "Mobile cloud computing: A survey." Future Generation Computer Systems 29.1, pp. 84-106, 2013.

[2] D. T. Hoang, C. Lee, D. Niyato, and P. Wang. "A survey of mobile cloud computing: architecture, applications, and approaches." Wireless communications and mobile computing 13, no. 18, pp. 1587-1611, 2013.

[3] S. F. Abedin, M. G. R Alam, Nguyen H. Tran, and C.S. Hong. "A Fog based system model for cooperative IoT node pairing using matching theory." In Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific, pp. 309-314. IEEE, 2015.

[4] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha. "The Role of Cloudlets in Hostile Environments."IEEE Pervasive Computing 12, no. 4, pp. 40-49, 2013.

[5] R. S. Sutton andA.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998

[6] Z. Zeyu, M.Li, X. Xiao, and J. Wang. "Coordinated resource provisioning and maintenance scheduling in cloud data centers." In INFOCOM, 2013 Proceedings IEEE, pp. 345-349. IEEE, 2013.

[7] F. Bonomi, R. Milito, J. Zhu, and 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, pp. 13-16. ACM, 2012.

[8] C. Yoo, "Network Architecture for LTE and Wi-Fi Internetworking", NMC consulting group, August 16, 2012. Available at: www.nmcgroups.com

[9] G. Weiß "Action selection and learning in multi-agent environments." InProceedings of the second international conference on From animals to animats 2: simulation of adaptive behavior: simulation of adaptive behavior, pp. 502-510. MIT Press, 1993.

[10] G. Weiß "Distributed reinforcement learning." In The Biology and Technology of Intelligent Autonomous Agents, pp. 415-428. Springer Berlin Heidelberg, 1995.

[11] L. Monica, R. Sethi, J. D. Ullman, and A. Aho. "Compilers: Principles, Techniques, and Tools.", 2006.

[12] J. Sztrik, "Basic Queueing Theory." University of Debrecen: Faculty of Informatics, 2011.

[13] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang. "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading." In INFOCOM, 2012 Proceedings IEEE, pp. 945-953. IEEE, 2012.

[14] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. "Clonecloud: elastic execution between mobile device and cloud." InProceedings of the sixth conference on Computer systems, pp. 301-314. ACM, 2011.

[15] B. Xiangping, J. Rao, and X. Cheng-Zhong, "Coordinated self-configuration of virtual machines and appliances using a model-free learning approach." Parallel and Distributed Systems, IEEE Transactions on 24, no. 4, pp. 681-690, 2013

[16] Do, Cuong T., Nguyen H. Tran, D. H. Tran, C. Pham, M. G. R. Alam, and C. S. Hong. "Toward service selection game in a heterogeneous market cloud computing." In Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on, pp. 44-52. IEEE, 2015.

[17] M. D. Kristensen, "Scavenger: Transparent development of efficient cyber foraging applications." In Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on, pp. 217-226. IEEE, 2010.