Least Loaded Sharing in Fog Computing Cluster

Sammy Chan
Department of Electronic Engineering
City University of Hong Kong
Hong Kong SAR
P.R. China
Email: eeschan@cityu.edu.hk

***Abstract*—Recently, there has been a growing ubiquity of connected devices and sensors in wireless sensor networks, health care systems, smart grids and smart cities, forming the Internet of Things (IoT). IoT devices generally have limited computation resources, and thus rely on the computational and storage resources of the cloud. However, IoT applications generally have a real-time requirement that cannot be fulfilled by mainstream cloud services. Therefore, a new paradigm called *fog computing* has emerged to offload the computation and storage needs of end user devices to the servers in the network edge. In this paper, we propose a least loaded sharing method to fully exploit the collaboration between fog servers and to achieve load balance among them. In our method, an overloaded server is able to react to temporary peaks of requests by forwarding the incoming requests to the least loaded neighbour server. The proposed method helps to reduce the blocking probability of requests and the delay experienced by accepted requests. We also develop a computationally efficient analytical model to evaluate the performance of our proposed method.**

***Keywords*–fog computing; load balancing; collaboration servers; buffer sharing.**

I. Introduction

Over the past decade, cloud computing has become a very popular computing paradigm [1]. By centralizing computing, storage, and network management functions in data centers, cloud computing has a high degree of polymerization of service computing. It enables end users to universally access on-demand computing services and frees them from the specification of many details. Entirely dependent on the Internet, cloud computing ensures the maximum utilization of computational resources by providing flexibility in the availability of data, software and infrastructure [2].

Recently, cloud computing is increasingly used to support Internet of Things (IoT) applications, due to the growing ubiquity of connected devices and sensors in wireless sensor networks, health care systems, smart grids and smart cities. Although cloud computing is renowned for its cost-effective and convenient service, it is encountering several challenges introduced by the emerging IoT. First, many IoT applications have a real-time requirement that cannot be fulfilled by mainstream cloud services [3][4]. Second, the vast and rapidly growing number of connected IoT devices inflate the amount of data generated at an exponential rate [5]. If all this data is sent to the cloud, prohibitively high network bandwidth would be required in the cloud system. Third, IoT devices generally have limited computation resources. Thus, they would not be able to fulfill the needs imposed by the IoT applications. Naturally, they could make use of the cloud by offloading computation tasks to it. However, it will be unrealistic and prohibitively expensive to support the interaction between the cloud and all those resource-constrained devices, as it involves complex protocols and resource-intensive processing.

To fill the technology gaps in supporting IoT, a new paradigm, *fog computing*, has been proposed. Fog computing emphasizes the network edge and distributes onerous tasks closer to end user devices [6][7]. As illustrated in Figure 1, fog computing extends cloud computing by bringing heterogeneous resources to the edge of the network, so that a substantial amount of data storage, computing and control functions, communication and networking is carried out near the end user. Besides, fog computing will not be faced with serious security issue as data travel from fog to end users within a short distance. Ultimately, the goals of fog computing are to reduce the data volume and traffic to cloud servers, offer low latency, and improve Quality of Service (QoS).

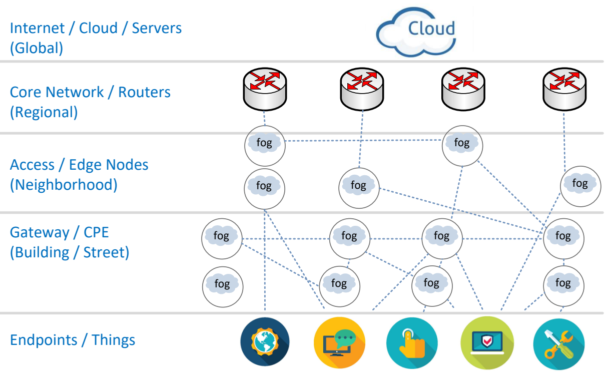

Figure 1. Architecture of fog computing.

Fog computing also shares many similar mechanisms and attributes with cloud computing [8]. For example, the core idea behind the two computing paradigms is to transfer load from end users to servers [9]. It means the problem of load imbalance is inevitable in both fog and cloud computing because requests of users arrive at servers randomly and frequently [10]. For the case of fog computing, servers with computing and storage capacities can be attached to the base stations of mobile telecommunication networks so that they are close to the end users. In the simplest case, an individual server only needs to execute tasks from its local users. In other words, each server operates independently. However, with the trend of deploying small cells, each server generally has some nearby neighbours. When a server experiences temporary overload, it could exploit the resources of its neighbours by forwarding newly arrived tasks to them. This is particularly attractive for fog computing, since it is designed to offer real-time service and thus needs to satisfy stringent QoS requirements, such as blocking of requests and service waiting time [11]. Under this situation where a set of servers collaborate with each other to serve users' requests, how to maintain load balance among the servers is an important issue.

In this paper, we propose a least loaded sharing method to fully exploit the collaboration between servers and achieve load balance among them. In our method, an overloaded server is able to react to temporary peaks of requests by forwarding the incoming request to the least loaded neighbour server. The proposed method helps to minimize the blocking probability of requests and reduce the delay experienced by accepted requests. We also develop a computationally efficient analytical model to evaluate the performance of our proposed method. The accuracy of the model is validated by computer simulations. Numerical results demonstrate that, by having servers share buffer space with each other, temporary load peaks can be efficiently relieved.

The remainder of the paper is organized as follows: Section II reviews some recent research work in optimization of computational resources in fog computing environments. Section III introduces our proposed least loaded scheme. Section IV presents the performance model of the scheme. Section V validates our model by simulation results and provides some numerical results to demonstrate the effectiveness of our proposed scheme. Finally, Section VI concludes the paper.

II. Related Work

Fog computing allows mobile terminals to have access to additional computational and storage resources, which are more abundant than those available in typical user equipment, by offloading demanding tasks to nearby fog servers. For the case when each individual fog server only serves its local users, research efforts have focused on the joint distribution of computational and radio resources for mobile terminals and the fog server. Early work covered the management aspects, the experimental evaluation of energy saving due to offloading, and the design of offloading criteria which consider the cost of radio resources of the access networks [12][13]. Since the optimal energy cost for the data transfer depends on the channel conditions, the Gilbert-Elliott channel model is used in [14] to study the radio-cloud interaction. The work provides some insights about how the quality of the wireless link affects the transmission rate and the offloading decision. In [15], a computation offloading distribution between a single user and the server is derived, taking into account the delay constraint of tasks and assuming that multiple antennas are available. Their results provide the optimal transmission strategy and the optimal distribution of the computational load between the user and the server. When the number of requests from users becomes very large, the computational resources of only one server sometimes may not be enough. In this situation, a number of servers can cooperate together through the formation of a cluster. This means that extra computational capacities can be provided to users. In [16], Barbarossa *et al.* consider a multi-user, multi-server and multi-cloud scenario in which the servers and cloud are organized in a different hierarchy. They study a joint optimization of computational and radio resources with the objective to minimize the power consumption of each user, subject to the latency constraints imposed by each user.

When fog servers form a cluster, how to improve the QoS delivered to users is also an important issue. In [10], a load balancing scheme between two fog servers is proposed to minimize the blocking probability at each server and the waiting time of the tasks. In this scheme, each server is assumed to have a buffer to store service requests from users for subsequent executions. When the buffer of a server is full, the newly arrived requests are forwarded to the neighbour server, which accepts the request only if its current queue length is below a given threshold. The system is modelled as a two-dimensional Markov chain to evaluate the performance of the proposed scheme. Numerical results demonstrate that both blocking probability and waiting time are reduced. The authors also propose a possible implementation of the load balancing scheme.

III. Least Loaded Sharing

We consider that, in an area of interest, there is a cluster of servers. Each server receives task requests from its local users, and executes the tasks on a first-come-first-serve basis. When a request arrives at a server and finds the server busy, it queues for its turn of service. Since the requests are expected to have strict delay requirements, there is a limit on the number of requests that can queue in a server. When the queue length of the server has reached the limit, for the case that each server operates independently, the request is blocked immediately to avoid excessive waiting time. On the other hand, when our least loaded sharing method is operated, the request is re-directed to the server with the shortest queue length. If there are multiple such servers, the request is re-directed to one of them randomly. As a result, a request is blocked only if the cluster of servers has reached the queue length limit.

IV. Performance Models

Let us assume that there are N servers, and task requests arrive at each server according to a Poisson process with rate λ tasks/second, as shown in Figure 2. The time needed to execute a task is exponentially distributed with mean $1/\mu$ seconds. The offered traffic to each server is given by $\rho = \lambda/\mu$. The limit on the queue length (including the one being served) is set to K. Here, two metrics are used to evaluate the performance of our proposed method. The first one is the average waiting time of tasks, $\overline{W}$, which is the average of the time from the arrival of a task at a server until the time that the server starts processing the task. The second one is the task blocking probability, p_B, which is the probability that a task is blocked by the fog computing system.

First, we present the performance model for the *no-sharing* case, i.e., the case in which each server operates independently. This case will be used for comparison in the next section. For each independent server, it can simply be modelled as a $M/M/1/K$ queueing system. Let π_i, $0 \leq i \leq K$, be the probability that there are i tasks in a server. For $\rho \neq 1$, it is well known that [18]

$$\pi_i = \rho^i \frac{1-\rho}{1-\rho^{K+1}}. \quad (1)$$

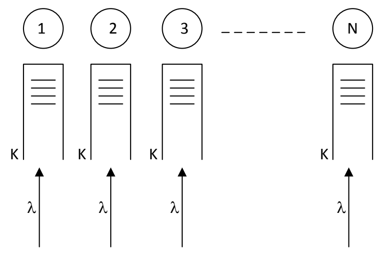

Figure 2. The queueing model of a cluster of fog servers.

The blocking probability is given by

$$p_B = \pi_K = \frac{\rho^K(1-\rho)}{1-\rho^{K+1}}, \tag{2}$$

and the mean waiting time is given by

$$\begin{aligned}\overline{W} &= \frac{\sum_{i=0}^{K} i\pi_i}{\lambda(1-\pi_K)} - 1/\mu \\ &= \left(\frac{1-(K+1-K\rho)\rho^K}{(1-\rho)^2} - 1\right)\frac{1}{\mu}.\end{aligned} \tag{3}$$

Next, we present the performance model for the least loaded sharing method. A server is said to be in state $i, 0 \leq i \leq K$, when there are i tasks in its buffer (including the one being served). Let p_i^j be the probability that server j is in state i. However, since the system under consideration is uniform, each server receives the same offered load and has the same state probabilities. Thus, p_i^j can be simplified as p_i. Consider a tagged server which is in state i and has the shortest queue length. The probability that there are $l-1$ other servers at state i is denoted as $F(l|i)$. Consider that a particular server is full. Its overflow rate to the tagged server at state i is given by

$$\begin{aligned} y_i &= \lambda p_K \sum_{l=1}^{N-1} \tfrac{1}{l} F\left(l/i\right) \\ &= \lambda p_K \sum_{l=1}^{N-1} \tfrac{1}{l} \tbinom{N-2}{l-1} {p_i}^{l-1} \left(\sum_{t=i+1}^{K} p_t\right)^{N-1-l} \end{aligned} \tag{4}$$

Let a_i be the total overflow rate of tasks to the tagged server when it is in state i,

$$a_i = (N-1)y_i \tag{5}$$

The tagged server can be modeled as a Markov chain, as shown in Fig 3. By using the local balance equation, we have

$$p_i = \frac{\lambda + a_{i-1}}{\mu} p_{i-1}, \quad i = 1, 2, \ldots, K \tag{6}$$

Therefore,

$$p_i = \frac{\prod_{j=0}^{i-1}(\lambda + a_j)}{\mu^i} p_0, \quad i = 1, 2, \ldots, K \tag{7}$$

Using the normalization condition $\sum_{i=0}^{K} p_i = 1$, p_0 is given by

$$p_0 = \left[\sum_{i=1}^{K} \frac{\prod_{j=0}^{i-1}(\lambda + a_j)}{\mu^i} + 1\right]^{-1}. \tag{8}$$

Equations (5) and (7) form a set of fixed-point equations. They can be solved by repeated substitution to obtain p_i.

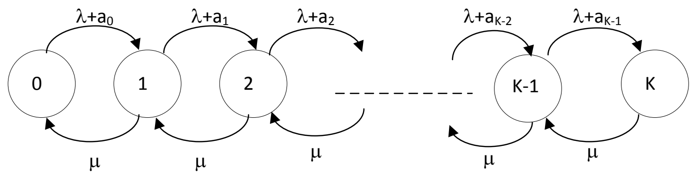

Figure 3. The Markov chain of a server.

By Little's theorem, $\overline{W}$ is given by

$$\overline{W} = \frac{\sum_{i=0}^{K} ip_i}{\lambda(1-p_K)} - 1/\mu \tag{9}$$

For p_B, at a first glance, one may think that it is given by $(p_K)^N$. However, the expression $(p_K)^N$ assumes that each server operates independently, which contradicts the fact that there is dependency among the servers. In order to obtain an exact value of p_B, we need to model the system as a N dimensional Markov chain with totally $(K+1)^N$ states, and then solve the state probabilities. Unfortunately, efficient methods for solving the state probabilities are available only for very small N. For example, in [10], the matrix-geometric approach of [17] is used for the case of $N = 2$. For large N or K, such an approach is analytically intractable. Here, we propose an approximate closed form solution for p_B. First, we assume that the buffers of individual servers are aggregated together. Then, the system can be modelled as a $M/M/N/NK$ system. However, such a model is more efficient than the actual system and thus under-estimates the blocking probability. Intuitively, we need to reduce the aggregated buffer size to reduce the amount of under-estimation. Since, on average, the number of tasks waiting in the server is $\frac{\overline{W}}{\mu}$, we postulate that the total buffer space available for sharing, K_{eff}, is given by $N(K-\frac{\overline{W}}{\mu}$. Therefore, we model the system as a $M/M/N/K_{eff}$ queue, and its blocking probability approximates p_B. Using the well established result of the blocking probability of a $M/M/N/k$ queue [18], p_B is given by

$$p_B = \pi_N \left(\frac{A}{N}\right)^{K_{eff}-N}, \tag{10}$$

where

$$\pi_N = \left(E_N^{-1}(A) + \rho\frac{1-\rho^{K_{eff}-N}}{1-\rho}\right)^{-1}, \tag{11}$$

$A = N\lambda/\mu$, and $E_N(A)$ is the Erlang B blocking probability for a $M/M/N/N$ queue with offered traffic A.

TABLE I. COMPARISON OF BLOCKING PROBABILITY OBTAINED BY $(p_K)^N$ AND SIMULATION

	Buffer size	K = 5	K = 10	K =15	K = 20	K = 25	K = 30
λ = 0.85	$(p_K)^N$	4.58E-05	8.56E-08	6.77E-10	8.48E-12	1.27E-13	2.06E-15
	simulation	1.17E-02	6.13E-04	3.36E-05	8.79E-07	5.06E-08	5.89E-10
λ = 0.90	$(p_K)^N$	1.76E-04	7.30E-07	1.56E-08	5.98E-10	3.05E-11	1.80E-12
	simulation	2.27E-02	3.27E-03	4.98E-04	7.12E-05	9.67E-06	1.31E-06

V. NUMERICAL RESULTS

In this section, we use the developed analytical model to evaluate the performance and assess the potential benefits of the least loaded sharing scheme under various parameters. At the same time, we validate the analytical model by simulation. For this purpose, we have built a discrete event simulator in C++ to generate simulation results. We set $\mu = 1$ second, and vary the arrival rate λ to obtain different loads. The duration of each simulation run varies according to the system parameters, ranging from 10^7 to 10^{10} seconds, but the warm-up period is fixed at 10^5 seconds.

First, we evaluate the blocking probability for $N = 5$, with various K and λ. Table I compares the blocking probabilities obtained by $(p_K)^N$ and simulation, for K varying from 5 to 30 with a step of 5, with $\lambda = 0.85$ and 0.9, respectively. It shows that $(p_K)^N$ under-estimates the blocking probabilities by several order of magnitudes, and thus justifies the need of a more accurate way to calculate the blocking probabilities. Figure 4 and Figure 5 illustrate the effect of buffer size on the blocking probability for two different loads. It can be seen that, when K increases, the blocking probability decreases exponentially. From the perspective of the validity of the proposed $M/M/N/K_{eff}$ model, the analytical results obtained by (10) are in good agreement with the simulation results. Furthermore, in comparison to the isolated scheme, the least loaded sharing scheme exhibits lower blocking probabilities. The reduction of blocking probabilities increases with buffer size. Clearly, this is because in the least loaded sharing scheme, the service and buffer capacity of all servers are aggregated together to serve the incoming tasks.

Figure 6 and Figure 7 depict the relationship between average waiting time experienced by tasks and the buffer size of each server in a fog computing system. It can be observed that the theoretical delay obtained by (9) is in good agreement with the results obtained from simulations. For both shared and isolated schemes, the mean delay of a task increases with the buffer size because more tasks are allowed to wait in the buffer. However, the delay incurred in the least loaded sharing scheme is always smaller than the isolated scheme. This is because, under the least loaded sharing scheme, tasks are more probable to enter a buffer with a shorter queue length.

It can be concluded that a single server working in isolation could reduce its blocking probability by simply increasing its buffer size. However, incoming tasks will suffer great system waiting time in such a system. On the other hand, the least loaded sharing scheme enables a fog computing system with heavy load and finite buffer size to offer low-latency processing of requests as well as low blocking probability.

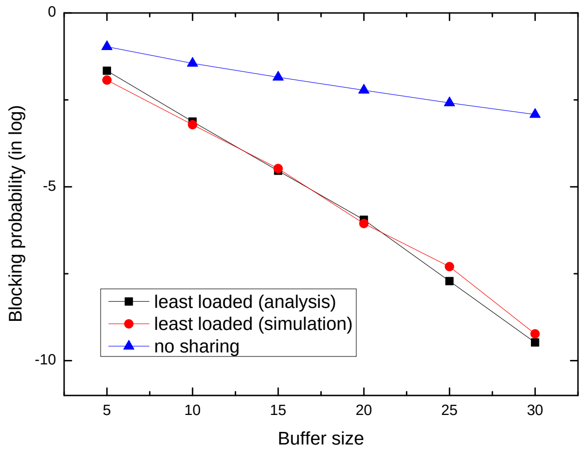

Figure 4. Blocking probability versus buffer size (λ = 0.85).

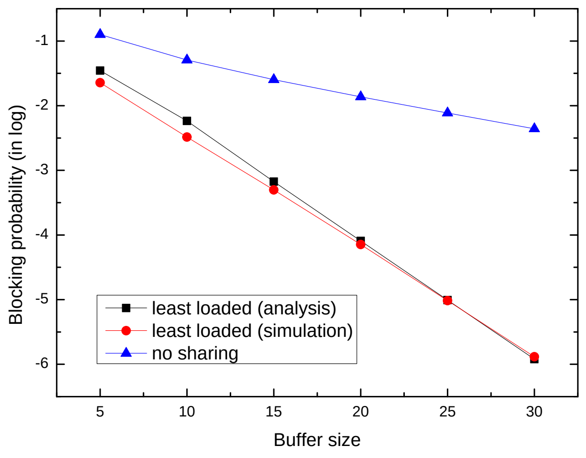

Figure 5. Blocking probability versus buffer size (λ = 0.9).

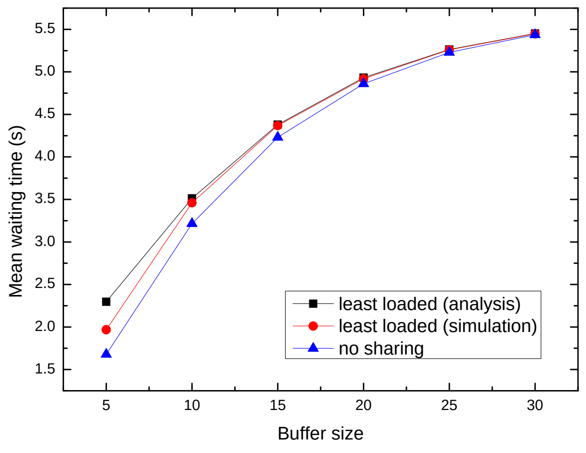

Figure 6. Mean delay versus buffer size (λ = 0.85).

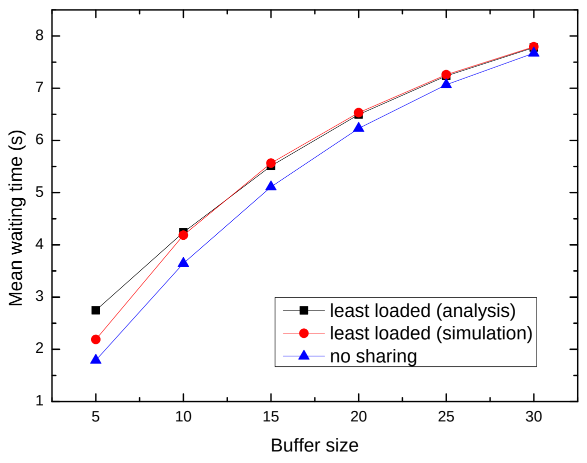

Figure 7. Mean delay versus buffer size (λ = 0.9).

VI. Conclusion

In this paper, we have proposed a least loaded sharing scheme for load balancing in a cluster of fog servers. We have developed an analytical model to evaluate the performance of the proposed scheme. The model is based on a state-dependent Markov chain. After solving the state probabilities of the Markov chain, the mean waiting time can be obtained. Also, a computationally efficient method has been developed to approximately calculate the blocking probability of requests. Simulation has been used to validate the model and show that the approximation is acceptable. Compared to the case when each server operates independently, our proposed scheme can utilize the resources of the cluster of fog servers more efficiently, leading to less waiting time and lower blocking probability experienced by users.

References

[1] A. Botta, W. D. Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: a survey", *Future Generation Computer Systems*, vol. 16, no. 3, 2014, pp. 1391-1412.

[2] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges", *Journal of internet services and applications*, 2010, vol. 1, no. 1, pp. 7-18.

[3] V. Sarathy, P. Narayan, and R. Mikkilineni, "Next generation cloud computing architecture: Enabling real-time dynamism for shared distributed physical infrastructure", *Proceedings, 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010*, pp. 48-53, 28-30 June, 2010, Greece.

[4] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K. R. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live VM migration framework", *IEEE Access*, 2017, vol. 5, pp. 8284-8300.

[5] R. Kelly. "Internet of Things Data to Top 1.6 Zettabytesby 2022", [Online]. Available: https://campustechnology.com/articles/2015/04/15/internet-of-things-data-to-top-1-6-zettabytes-by-2020.aspx [retrieved: April, 2019].

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things", *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13-16.

[7] N. Peter, "Fog computing and its real time applications" *International Journal of Emerging Technology and Advanced Engineering*, vol. 5, no. 6, pp. 266-269, 2015.

[8] X. Masip-Bruin, E. Marin-Tordera, G. Tashakor, A. Jukan, and G. Ren, "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems", *IEEE Wireless Communications*, vol. 23, no. 5, pp. 120-128, November 2016.

[9] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption", *IEEE Internet of Things Journal*, 2016, vol. 3, no. 6, pp. 1171-1181.

[10] R. Beraldi, A. Mtibaay, and H. Alnuweiri, "Cooperative Load Balancing Scheme for Edge Computing Resources", *Proceedings, 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC 2017)*, Valencia, Spain, 8-11 May, 2017, pp. 94-100.

[11] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing", *Proceedings, IEEE 81st Vehicular Technology Conference (VTC Spring)*, 2015, pp. 1-6.

[12] L. Gkatzikis and I. Koutsopoulos, "Migrate or not? Exploiting dynamic task migration in mobile cloud computing systems," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 24-32, June 2013.

[13] K. Kumar, J. Liu, Y. H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129-140, February 2013.

[14] W. Zhang *et al.*, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569-4581, September 2013.

[15] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. on Vehicular Technology*, vol. 60, no. 10, pp. 4738-4755, October 2015.

[16] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo,"Joint allocation of computation and communication resources in multiuser mobile cloud computing," *Proceedings, 2013 IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp.26-30, 16-19 June 2013, Darmstadt, Germany.

[17] E. H. Elhafsi and M. Molle, "On the solution to QBD processes with finite state space", *Stochastic Analysis and Applications*, vol. 25, no. 4, pp. 763-779, 2007.

[18] M. Zukerman, *Introduction to Queueing Theory and Stochastic Teletraffic Models*, 2014. [Online]. Available: http://www.ee.cityu.edu.hk/zukerman/classnotes.pdf [retrieved: April, 2019].