Mobile Edge Computing Performance Evaluation using Stochastic Petri Nets

Carvalho, Daniel; Rodrigues, Laecio; Takako Endo, Patricia; Kosta, Sokol; Airton Silva, Francisco

Published in:
2020 IEEE Symposium on Computers and Communications (ISCC)

DOI (link to publication from Publisher):
10.1109/ISCC50000.2020.9219650

Publication date:
2020

Document Version
Accepted author manuscript, peer reviewed version

Mobile Edge Computing Performance Evaluation using Stochastic Petri Nets

Daniel Carvalho[∓], Laécio Rodrigues[∓], Patricia Takako Endo[⋆], Sokol Kosta[◦] and Francisco Airton Silva[∓]
∓ Universidade Federal do Piauí (UFPI), Brazil
⋆ Universidade de Pernambuco (UPE), Brazil
◦ Aalborg University Copenhagen, Denmark
E-mail:faps@ufpi.edu.br, patricia.endo@upe.br, sok@es.aau.dk

***Abstract*—Mobile Edge Computing (MEC) is a network architecture that takes advantage of resources available at the edge of the network to enhance the mobile user experience by decreasing the service latency. MEC solutions need to dynamically allocate the requests as close as possible to their users to avoid high latency. However, the request allocation does not depend only on the geographical location of the servers, but also on their requirements. The task of choosing and allocating appropriate servers in a MEC environment is challenging because it involves many parameters. This paper proposes a Stochastic Petri Net (SPN) model to represent a MEC scenario and analyze its performance. The model focuses on parameters that can directly impact the service Mean Response Time (MRT) and resource utilization level. We propose case studies with numerical analyzes using real-world values to validate the proposed model. The main objective is to provide a practical guide to assist infrastructure administrators to adapt their architectures, finding a trade-off between MRT and resource utilization level.**

***Index Terms*—Mobile Edge Computing, Internet of Things, Stochastic Models.**

I. Introduction

According to the Statista (a German online portal for statistics), the number of mobile devices will be around 16.8 billion in 2023 [1]. This significant adoption is fundamentally driven by the increase of new mobile users as well as the development of new interactive applications [2] [3]. However, the increasing number of mobile devices has an extreme effect on mobile networks, bringing significant challenges for telecommunications companies [4]. Mostly, mobile devices have limited computational capacity (processing, storage, and battery), and cellular networks are characterized by high power consumption, low bandwidth, and high latency [5]. Besides, the exponential growth of the Internet of Things (IoT) promises to make wireless networks even more congested [6]. To cope with these challenges, many researchers have proposed optimized architectures for this context, such as Mobile Cloud Computing (MCC) [7, 8].

MCC is the integration of cloud computing and mobile computing, which provides additional capabilities for mobile devices by centralizing their resources at the cloud infrastructure [9]. However, the Internet of Things scenario has brought harder challenges to MCC, such as super high latency, security vulnerability, and limited data transmission, because cloud resources are often far away from the end users [10].

As an evolution of the MCC architecture, Mobile Edge Computing (MEC) has emerged. The main goal of MEC is to address the challenges that MCC has been facing, deploying resources closer to the end-users, allowing computing and storage operations to be performed nearby the source device [11]. The MEC concept has emerged to unite telecom operators, IT, and cloud computing companies to deliver cloud services directly at the network edge. Differently from traditional cloud computing systems where remote public clouds are used, MEC servers are usually owned by the network operator. They are implemented directly at the cellular tower or the local wireless Access Points (APs) using a generic-computing platform. With this position, MEC allows the execution of applications near the end-users, substantially reducing the end-to-end delay and releasing the burden on backhaul networks.

Since MEC is still considered a recent topic, some research opportunities are open to being explored, such as performance evaluation of MEC architectures [12]. Analytical models are suitable to evaluate the performance of complex systems, mainly during the initial stages. Stochastic modeling is a popular formal method for performance evaluation in concurrent systems with synchronization and communication mechanisms. Stochastic Petri Nets (SPNs) are special cases of stochastic models [13], that are enabled to set upstate equations, algebraic equations, and other mathematical models governing the behavior of systems. The use of SPNs has already been successfully applied in the context of MCC in previous works [14, 15]. However, to the best of our knowledge, performance analytical models in MEC architectures are scarce until the present moment.

In this paper, we focus on the MEC server performance evaluation in an environment that provides wireless Internet coverage for mobile users in a large-scale metropolitan area. For that, we propose an SPN model to represent the MEC architecture and evaluate the trade-off between *mean response time (MRT)* and *resource utilization*. The *MRT* is defined as the time duration for a request to be processed by MEC servers and the result to be returned to the mobile client device. The *resource utilization*, as the name implies, indicates the percentage of resources that are on active mode. In other words, resource utilization is related to the number of computers (physical machines/VMs/containers) or CPU cores

that are being used.

The main contributions of this paper are: *(i)* an analytical model, which serves as a useful tool for system administrators to evaluate the performance of different MEC architectures, especially during the planning phase before deployment; *(ii)* a case study with real data, which serves as a practical guide for the performance analysis in MEC architectures.

The remainder of this paper is organized as follows: Section II discusses the main related works; Section III presents the MEC architecture that was used as the basis for the construction of the model; in Section IV we present the SPN model with respective metrics and case studies; and Section VI traces some conclusions and future works.

II. Related work

Some related works propose MEC architectures in **cooperation with other layers**. However, these papers do not exploit different configurations of computational resources. *Tong et al.* propose to deploy cloud servers at the edge of the network and allocate the servers in geographically distributed hierarchies to use the cloud to meet the peak loads of requests coming from MEC [16]. *Chen et al.* focus on an optimization algorithm to calculate where it will process the workload (MEC or cloud). The authors use generic applications that require a high level of processing to validate their solution [17]. Authors in [18] focus on the MEC and cloud's ability to process large data loads. The main objective of the paper is to try to reduce the latency between the two layers by assigning weights to the complexities of the tasks to be performed remotely.

Some papers focus on the problem of **energy consumption of mobile devices** with the aid of a MEC architecture. *Zhang et al.* have studied *offloading* mechanisms investigating a MEC architecture and 5G networks [19]. The authors have formulated an optimization problem to minimize energy consumption by observing processing time and data transfer. *Trinh et al.* have studied the potential of MEC to mitigate the battery limitation of IoT devices [20]. The authors have used a facial recognition application to demonstrate the feasibility of offloading decision policies. *Mao et al.* have also proposed a battery optimization algorithm on mobile devices [21]. The algorithm includes frequencies of the MEC server, CPU cycle, and latency rate. An advantage of this algorithm is that decisions depend only on instantaneous server-side information without information about the distribution of the task requests.

Unlike these studies, our paper does not directly analyze the energy expenditure of client devices. However, the MRT metrics adopted here are proportionally related to the energy expenditure of client devices. The longer the request takes to execute, the higher the energy consumption.

The closest related works are those that deal with **MEC infrastructure planning**. *Premsankar et al.* have carried out a real experimental evaluation with a high interactivity electronic game, but with the limitation of having only one mobile client [22]. The work of *Jararweh et al.* developed an architecture simulator that included a Cloudlet layer and a MEC layer [10]. The authors try to increase the coverage area for mobile users, where users can make requests with minimum costs in terms of energy expenditure. The authors only have adopted the number of requests as an input parameter and the use of a custom simulator without detailed explanations about its characteristics. *Liu et al.* have adopted a Markov chain model as a decision process on where tasks should be performed (locally or on the MEC server) [23]. The model takes into account the queuing status of the task buffer, the state of the mobile device execution, and the network state. However, the authors do not consider the MEC server with multiple parallel nodes, and thus not taking advantage of the potential of server parallelism. *Badri et al.* also used Markov chains to decide where to execute requests on multiple MEC towers [24]. The algorithm takes into account the movement of users, the cost of communication between users and servers, the cost of running each server, and the allocation cost.

Unlike our proposal, the works mentioned above do not exploit the resource utilization, and only some consider the MRT metric. All the papers are limited in terms of parameterization of the evaluations, with at most observing three architecture configuration parameters. None of the papers (except [22]) have addressed applications with a high level of interaction. Besides, none of the papers has adopted Stochastic Petri nets.

III. MEC Architecture

This paper presents an SPN to model a MEC architecture where resources of single servers are parallelized using containers, for example. The main objective is to minimize resource costs and maximize performance. Figure 1 illustrates the MEC architecture we are considering for performance modeling and analysis. The architecture is composed of three parts: **Mobile Devices**, **FrontEnd**, and **Edge Computing**. The mobile devices request services to the FrontEnd, and the FrontEnd distributes the requests among the edge computing.

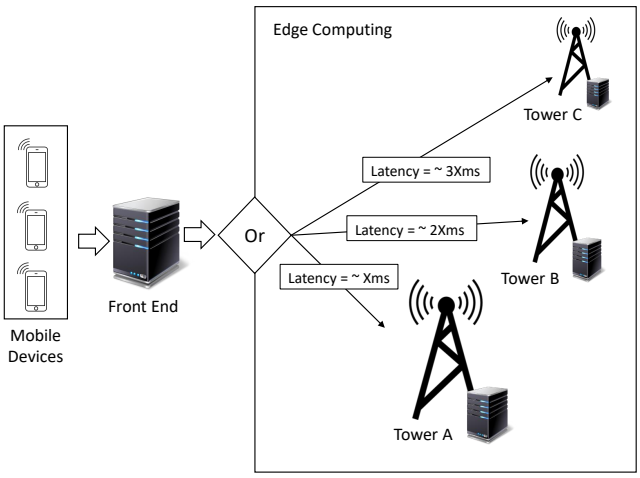

Fig. 1. Adopted MEC architecture for performance evaluation

The data streams are generated by applications running on the **Mobile Devices**. These applications can range from health monitoring applications [25] to gaming applications [26], for

example. Therefore, servers proximity and high computing power are mandatory to provide efficient resource availability and low mean response time. The **FrontEnd** is responsible for receiving requests from the mobile devices and forwarding them to the servers. For that, it is responsible for choosing which server is best suited to perform the request. In this work, for taking this decision, we are considering two different parameters: *(a)* each server can have different configurations (e.g., number of cores), and *(b)* the distance from the tower to the mobile devices, which impacts on the server placement decision. In the **Edge Computing** layer, we are considering MEC servers with distinct latencies (Server A, Server B, and Server C, as shown in Figure 1). Each server has a set of slave nodes that can be, for example, containers running microservices. We consider that each container runs in a single server core. Even though virtual machines have been highly adopted in the MEC research field [27], containers allow greater flexibility to scale the computing power according to the dynamic demand.

Given this architecture, the main objective of this work is to propose an SPN to evaluate the performance of different configurations of a MEC environment.

IV. SPN Base Model

An SPN model is useful when a service manager wants to plan and analyze changes in the system before implementing them. Figure 2 presents our proposed SPN model composed of three main blocks (highlighted in red): *(i)* **Device block**: responsible for generating user requests; *(ii)* **FrontEnd block**: represents the FrontEnd server and is responsible for receiving the user requests and forwarding them to one of the available servers. Load balancing policies can be applied at this point. In this work the requests can be forwarded to any of the available servers; *(iii)* **Edge Computing block**: represents the servers (A, B, and C) that are responsible for receiving and processing the requests, distributing the data among containers.

The *Admission* block consists of two places, *P_Arrival* and *P_ArrivalQueue*. *P_Arrival* represents the generation of user requests and *P_ArrivalQueue* represents the acceptance of these requests in the queue. The transition *T_Arrival* is configured with the arrival request rate. In this model, we consider the arrival time to be exponentially distributed, but it can be easily modified to fit other distributions. The transition *ND* (network delay) represents the sending and receiving times of the request to the FrontEnd server. *ND* fires as soon as there is a token at *P_ArrivalQueue* and at least one token at *P_FrontEndCapacity*.

When the transition *ND* fires, the block *FrontEnd* is reached, and one token is consumed from places *P_ArrivalQueue* and *P_FrontEndCapacity* and one token is produced at places *P_FrontEndInProcess* and *P_Arrival*. The amount of tokens at *P_FrontEndInProcess* represents the request queuing at *FrontEnd*. The queue happens because there are no available resources on the servers. When the request processing starts, the tokens are consumed from the *P_FrontEndInProcess*. If there are available resources in one of the three servers, one of the transitions *T_A*, *T_B*, or *T_C* fires and the request follows the path to be processed.

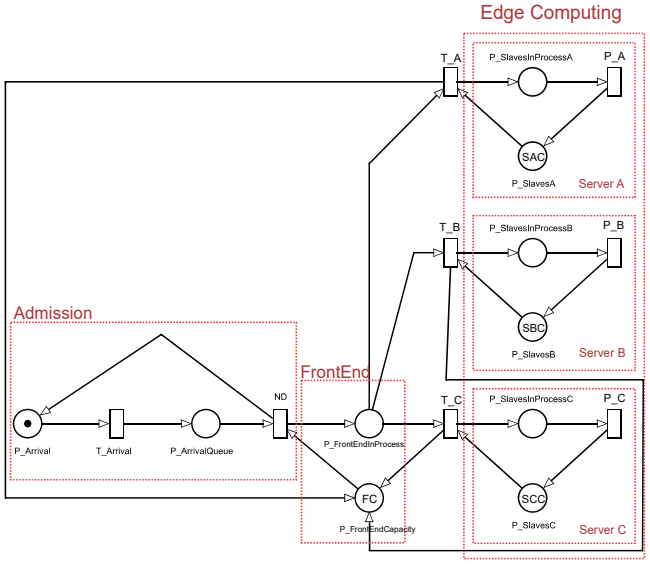

Fig. 2. SPN model for MEC architecture

The transitions *T_A*, *T_B*, and *T_C* represent the transfer time from the *FrontEnd* to the respective MEC servers. Considering the Server A, the transition *P_A* represents the beginning of the processing. The *P_A* fire is conditioned to the number of available nodes to process requests at *P_SlavesA* (marking SAC).

Given the proposed model, it is possible to configure 15 parameters, giving the possibility to run a large number of different scenarios. The configurable parameters are eight timed transitions and four places related to resource capacities. Any configuration of these parameters impacts substantially on the mean service time and consequently, on the infrastructure cost. The possibility of running a high number of scenarios is one of the main contributions of this work. Note that the proposed SPN covers only three MEC servers, but it is easily expandable to represent more servers.

A. Performance metrics

In this work, we consider two metrics: the mean response time (MRT), which refers to the time required for a request to be executed on the system, and the utilization level that calculates the resource consumption of the computing nodes.

The *MRT* can be obtained from the Little's Law [28], which relates three variables: the mean number of requests in progress in a system, $RequestsInProcess$, the arrival rate of new requests, ARR, and the mean response time, MRT. The arrival rate is the inverse of the arrival delay, AD. Little's Law requires a stable system, meaning that the arrival rate is lower than the service time. Therefore, MRT is calculated using the equation: $MRT = \frac{RequestsInProcess}{ARR}$.

The number of requests in progress is obtained by summing up the number of tokens in each place that means work in progress (e.g.: *P_FrontEndInProcess* and

P_SlavesInProcessA). This number of requests is obtained by the probabilistic expected number of requests in each place ($Esp(Place)$).

The percentage of resource usage for each machine can be obtained by the following equations: $U(A) = \frac{Esp(P\_SlavesInProcessA)}{SAC}$, $U(B) = \frac{Esp(P\_SlavesInProcessB)}{SCB}$ and $U(C) = \frac{Esp(P\_SlavesInProcessC)}{SCC}$. The resource usage is calculated by the number of tokens in the queue that are consuming resources from the server divided by the total resource capacity of that given server. In other words, the usage of resources is the number of containers that are being used. Given the utilization per server, we are also able to calculate the usage of all available resources in the system by using equation: $U(all) = \frac{U(A)+U(B)+U(C)}{n\_servers}$.

B. Case Study

The main contribution of our proposed model is the versatility and flexibility to evaluate different scenarios. So, to show the applicability of our proposed model, we present some numerical analyzes that exemplify how our model can be used effectively by a system administrator. We have considered the system parameters used in [22] as input parameters for our model. The authors evaluated a MEC architecture with a single mobile device as a client and containers executing the services. Authors have evaluated a 3D game called *Neverball*, where the player must tilt the floor to control the ball to collect coins and reach an exit point before the time runs out. Therefore, our study evolves the work in [22] by performing numerical analysis to evaluate the scenarios considering multiple parameters. In our scenarios, the MEC architecture is composed of three servers, and the machines available for evaluation have 8, 16, or 32 cores. We consider one of the scenarios in [22] with a game resolution of 800×600 pixels. The adopted parameter value corresponding to the processing delay of a request is 10.7ms. All used parameters were: T_A = 8.3, T_B = 5.2, T_C = 2.5, P_A = 10.7, P_B = 10.7, P_C = 10.7, ND = 2.3.

Therefore, the goal of this case study is to minimize the MRT and efficiently allocate tasks to machines that do not present a high resource consumption rate to the game Neverball. The aim is to minimize the deployed resources while still fulfilling the requested demand to lower the financial investments. We assume that the desired MRT is up to 40ms, and the desired usage rate should be between 40% and 50% to avoid overload and idleness in servers. In this case, the request arrival rate is 0.05 requests per milliseconds. The list of all 27 possible servers' configurations is presented in Table I.

Figure 3 presents the results. Configuration #1 presents a setup with MRT equals to 995.82ms, which is far above the others, so we have omitted this configuration to provide better visualization. The configurations that satisfy the requirements (MRT and resource utilization levels) are #27, #18, #24, #15, #9, #6, #26, #21, #12, #7, #23, #3, #14 and #8. The best configuration is #27, which is composed of the best available servers (all of them with 32 cores), and the second-best setting (ServerA=16, ServerB=32, ServerC=32) is focused on giving more computation power to servers that are far from the user.

TABLE I
POSSIBLE CONFIGURATIONS FOR CASE STUDY 1 AND 2.

Configuration	Server A	Server B	Server C
#1	8	8	8
#2	8	8	16
#3	8	8	32
#4	8	16	8
#5	8	16	16
#6	8	16	32
#7	8	32	8
#8	8	32	16
#9	8	32	32
#10	16	8	8
#11	16	8	16
#12	16	8	32
#13	16	16	8
#14	16	16	16
#15	16	16	32
#16	16	32	8
#17	16	32	16
#18	16	32	32
#19	32	8	8
#20	32	8	16
#21	32	8	32
#22	32	16	8
#23	32	16	16
#24	32	16	32
#25	32	32	8
#26	32	32	16
#27	32	32	32

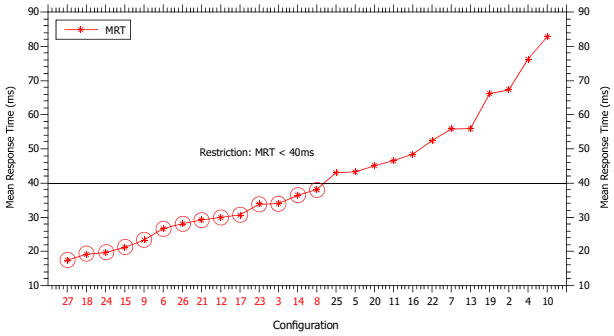

Fig. 3. Results considering MRT $\leq$ 40ms.

Figure 4 presents the resource utilization of servers' configurations. Now, the configurations that fit the requirement are #25, #14, #8, #3, #16, #20, #22 and #5. In this case, the best solution (ServerA=32, ServerB=32, ServerC=8) is based on machines with more computation power closer to users. So, analysing the MRT and resource utilization results, one can note that the configurations #14 (ServerA=16, ServerB=16, ServerC=16), #8 (ServerA=8, ServerB=32, ServerC=16), and #3 (ServerA=8, ServerB=8, ServerC=32) fit both requirements. In configuration #14, the solution balances the computational power equally, but in configuration #8 and #3, the solutions are heterogeneous, allocating more computational power on servers that are further from the user.

V. SPN VALIDATION

To validate our proposed SPN model, we developed a prototype to compare the MRT calculated by the model against the MRT collected through real experiments. We simulate

TABLE II
RESULTS OBTAINED WITH THE ONE SAMPLE T-TEST

Number of Requests	Sample Number	MRT Mean (ms) - Experiment	MRT Mean (ms) - Model	Confidence Interval	Standard Deviation	P-Value
3	15	1116.5	1103.1	[1048.8 - 1184.1]	122.1	0.676
6	15	2437.8	2504.2	[2238.4 - 2637.2]	360	0.488
9	15	4398.3	4457.4	[4168.5 - 4628.2]	415	0.59

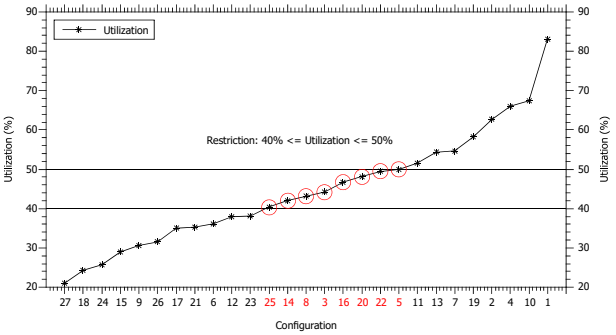

Fig. 4. Results considering 40% $\leq$ resource utilization $\leq$ 50%.

three towers with one server each. Five computers with the configuration of *Intel Core i3 8GB* were used in the validation. The servers are quad-core; each core runs an independent container. The requests are distributed among the containers following a round-robin scheme. A synthetic system was developed to send requests obeying an exponential distribution.

The execution times are collected at the beginning and end of each task. A local server plays the role of a global clock. Each request used in the experiment consists of a file containing a 10000x10000 matrix filled with random values. When the input file is sent, the machine requests the global clock server for the timestamp and saves it in a log. When the data is received on another server, it performs the same action. With these two timestamps, we get the total execution time. From these values, the *T_A, T_B*, and *T_C* transitions were populated with FrontEnd's transmission times for servers A, B, and C. In the *P_A, P_B* and *P_C* transitions, we add processing times for each server.

The request generator simulates a fixed amount of clients sending requests at the same time, FrontEnd receives all requests simultaneously, distributing them among the servers and consequently to the containers. To make the result more reliable, for each server configuration, we repeat the execution of the experiment 15 times. Considering the numbers of requests (3, 6, and 9), the MRTs obtained with the SPN model were respectively 1103.1ms, 2504.2ms, and 4457.4ms.

We applied the one-sample T-Test[1] to compare the MRT generated by the model with the MRT obtained in the experiments. All samples presented normal distribution. To verify the T-Test significance, we observe the p-value. Table II summarizes the results. The p-value is higher than 0.05 in all cases. Therefore, we cannot refute the null hypothesis of equality in all cases with 95% of confidence. Therefore, we can conclude that the results generated by the model are statistically equivalent to the experiment. The model reflects the real environment and is useful for planning MEC architectures.

[1]Sample T-Test https://tinyurl.com/yanthw4e

VI. CONCLUSIONS AND FUTURE WORKS

This paper proposed an SPN model to represent and evaluate the performance of a MEC architecture composed of n servers. Our model allows estimating the Mean Response Time (MRT) and the resource utilization level at the edge of the network. One of the most important aspects of our model is its evaluation flexibility due to the high number of parameters that can be configured by the user (15 parameters), making the tests more active and decisive. Numerical analyses were performed using real data collected from a reference work to show the applicability of our proposed model. Through the numerical analyzes, we could observe the MRT behavior and resource utilization, demonstrating the usefulness of the model in choosing the best way to implement a MEC architecture.

As future work, we plan to extend the SPN model to include availability metrics and measure energy. Given the extended model, we plan to perform new numerical analyzes using different scenarios.

ACKNOWLEDGEMENT

This research is funded by the National Council for Scientific and Technological Development (CNPq) and the company Virtex Telecom.

REFERENCES

[1] Statista forecast. https://www.statista.com/statistics/245501/multiple-mobile-device-ownership-worldwide/. Accessed: 2020-02-14.

[2] Stojan Kitanov, Edmundo Monteiro, and Toni Janevski. 5g and the fog—survey of related technologies and research directions. In *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pages 1–6. IEEE, 2016.

[3] Michael Till Beck, Martin Werner, Sebastian Feld, and S Schimper. Mobile edge computing: A taxonomy. In *Proc. of the Sixth International Conference on Advances in Future Internet*, pages 48–55. Citeseer, 2014.

[4] Eleonora Cau, Marius Corici, Paolo Bellavista, Luca Foschini, Giuseppe Carella, Andy Edmonds, and Thomas Michael Bohnert. Efficient exploitation of mobile edge computing for virtualized 5g in epc architectures. In *2016 4th IEEE international conference on mobile cloud computing, services, and engineering (MobileCloud)*, pages 100–109. IEEE, 2016.

[5] Gabriel Orsini, Dirk Bade, and Winfried Lamersdorf. Computing at the mobile edge: Designing elastic android applications for computation offloading. In *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 112–119. IEEE, 2015.

[6] Eleonora Borgia, Raffaele Bruno, Marco Conti, Davide Mascitti, and Andrea Passarella. Mobile edge clouds for information-centric iot services. In *2016 IEEE symposium on computers and communication (ISCC)*, pages 422–428. IEEE, 2016.

[7] Marcelo Antonio Marotta, Leonardo Roveda Faganello, Matias Artur Klafke Schimuneck, Lisandro Zambenedetti Granville, Juergen Rochol, and Cristiano Bonato Both. Managing mobile cloud computing considering objective and subjective perspectives. *Computer Networks*, 93:531–542, 2015.

[8] K. Sucipto, D. Chatzopoulos, S. Kosta±, and P. Hui. Keep your nice friends close, but your rich friends closer — computation offloading using nfc. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.

[9] Hoang T Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, 13(18):1587–1611, 2013.

[10] Yaser Jararweh, Ahmad Doulat, Omar AlQudah, Ejaz Ahmed, Mahmoud Al-Ayyoub, and Elhadj Benkhelifa. The future of mobile cloud computing: integrating cloudlets and mobile edge computing. In *2016 23rd International conference on telecommunications (ICT)*, pages 1–5. IEEE, 2016.

[11] Yaser Jararweh, Ahmad Doulat, Ala Darabseh, Mohammad Alsmirat, Mahmoud Al-Ayyoub, and Elhadj Benkhelifa. Sdmec: Software defined system for mobile edge computing. In *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, pages 88–93. IEEE, 2016.

[12] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili. Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine*, 55(4):54–61, April 2017.

[13] A.A. Desrochers, R.Y. Al-Jaar, and IEEE Control Systems Society. *Applications of petri nets in manufacturing systems: modeling, control, and performance analysis*. IEEE Press, 1995.

[14] Francisco Airton Silva, Sokol Kosta, Matheus Rodrigues, Danilo Oliveira, Teresa Maciel, Alessandro Mei, and Paulo Maciel. Mobile cloud performance evaluation using stochastic models. *IEEE Transactions on Mobile Computing*, 17(5):1134–1147, 2018.

[15] Thiago Pinheiro, Francisco Airton Silva, Iure Fe, Sokol Kosta, and Paulo Maciel. Performance and data traffic analysis of mobile cloud environments. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4100–4105. IEEE, 2018.

[16] Liang Tong, Yong Li, and Wei Gao. A hierarchical edge cloud architecture for mobile computing. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[17] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24(5):2795–2808, 2016.

[18] Jiping Wang Guangshun Li and Jianrong Song Junhua Wu. Data processing delay optimization in mobile edge computing. 2018, 2018.

[19] Mao Y Zhang Ke, Zhao Q Leng S, and Peng X Li L. Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks. 4:5896–5907, 2016.

[20] Chemodanov Trinh and Lei Yao. Energy-aware mobile edge computing for low-latency visual data processing. pages 128–133, 2017.

[21] Yuyi Mao, Jun Zhang, and Khaled B Letaief. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590–3605, 2016.

[22] Mario di Francesco Gopika Premsankar and Tarik Taleb. Edge computing for the internet of things: A case study. 5:1275–1284, 2018.

[23] Juan Liu, Yuyi Mao, Jun Zhang, and Khaled B Letaief. Delay-optimal computation task scheduling for mobile-edge computing systems. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1451–1455. IEEE, 2016.

[24] Hossein Badri, Tayebeh Bahreini, Daniel Grosu, and Kai Yang. Multi-stage stochastic programming for service placement in edge computing systems: poster. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, page 28. ACM, 2017.

[25] Amir M Rahmani, Tuan Nguyen Gia, Behailu Negash, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, and Pasi Liljeberg. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, 78:641–658, 2018.

[26] Xu Zhang, Hao Chen, Yangchao Zhao, Zhan Ma, Yiling Xu, Haojun Huang, Hao Yin, and Dapeng Oliver Wu. Improving cloud gaming experience through mobile edge computing. *IEEE Wireless Communications*, 2019.

[27] L. Zhao, J. Liu, Y. Shi, W. Sun, and H. Guo. Optimal placement of virtual machines in mobile edge computing. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, Dec 2017.

[28] John DC Little. A proof for the queuing formula: L= λ w. *Operations research*, 9(3):383–387, 1961.