A PSO Model with VM Migration and Transmission Power Control for Low Service Delay in the Multiple Cloudlets ECC Scenario

Tiago Gama Rodrigues*, Katsuya Suto†, Hiroki Nishiyama*, and Nei Kato*
* Graduate School of Information Sciences, Tohoku University, Sendai, Japan
Emails: {tiago.gama.rodrigues, bigtree, kato}@it.is.tohoku.ac.jp
† Waterloo University, Waterloo, Canada
Email: k.suto@ieee.org

***Abstract*—Mobile devices are naturally limited due to their portable sizes and will therefore never be equal to their desktop counterparts. To overcome this, Edge Cloud Computing can be utilized to execute tasks on behalf of the devices, allowing them to run applications that would normally be too demanding. In this service model, it is important to maintain a low Service Delay to keep the service transparent to the user. This can be achieved by focusing on lowering the Transmission Delay and Processing Delay. While existing approaches in the literature focus on one of those two, we postulate that only when considering both delays you can efficiently lower Service Delay and provide quality to all applications. In order to do that while being feasible, we propose a method based on Particle Swarm Optimization for lowering Service Delay in Edge Cloud Computing. Our proposal is shown to be close to optimality while still maintaining a low execution time for multiple cloudlets scenarios. Moreover, our proposal outperforms existing approaches from the literature with single focus on computation or communication, even in situations with high processing and transmission burdens, proving the superiority of a dual focus approach.**

I. Introduction

Recent surveys show that, while the amount of desktop environments has remained constant, the number of mobile devices is rising; moreover, this pattern is predicted to continue for many years [1]. This is an issue because mobile devices will never measure up to their desktop counterparts in terms of capability [2], [3], since the reason they are popular, their portability, results in a limited size and restrictions in processing, battery, storage, among others [4], [5]. Despite that, users still want to execute the applications from their desktop devices, meaning a solution to those limits must be developed. Enter Edge Cloud Computing (ECC), an edge network technology where users are paired with small scale cloud servers (called cloudlets): any tasks too demanding created by the users are sent wirelessly to the cloudlet, which hosts a Virtual Machine (VM) server that executes the task and sends the output wirelessly to the user (it is noteworthy that each VM server services only one user, which is convenient for migrating and customizing the VM). This whole process is transparent to the user, as if the mobile device was running the application locally [3], [6]–[11], and allows the devices to execute applications that normally require too much of them.

ECC must have low Service Delay as to not break the transparency and maintain a high Quality of Service (QoS) [8]. That can be divided into two goals: decreasing the time transmitting input/output (Transmission Delay), and decreasing the time computing the task (Processing Delay) (while there are other delays involved, those two are the only ones in effect every time a task is issued, dominating in long timescales). Most approaches in the literature focus on a single one of those goals [8], [12]–[14]. However, this way is not the best for two reasons [11]: firstly, there is a wasted potential for lowering the Service Delay when you only improve one of the delays; secondly, focusing on one of the delays is detrimental for applications that have a higher need of the other delay. As far as we know, the single paper that integrates both delays considers only a limited scenario with 2 cloudlets [11].

In this paper, we propose a method for lowering both Processing Delay and Transmission Delay in a scenario with an undetermined amount of cloudlets. The proposal utilizes VM migration and transmission power control as its main technologies for achieving its goals: effectively lower Service Delay as much as possible, provide a high QoS for various application profiles, and stay computationally feasible. We utilize a mathematical model with a Particle Swarm Optimization (PSO) model [15], an Artificial Intelligence technique, to achieve a low execution time and high efficiency.

II. Related Works

In our assumed scenario, Transmission Delay is defined as the time spent transmitting data between the user and the cloudlet; this means the interval between the user sending its task input and the cloudlet receiving it as well as the interval between the cloudlet sending the output and the user receiving it. Meanwhile, the Processing Delay is located in the middle of those intervals, where the task, after arriving to the corresponding VM server hosted by the cloudlet, stays in a queue, waiting for one of the server's processors to be idle, and later gets a hold of the processor resource and is executed. Fig. 1 contains a diagram depicting those two delays. Each task produced by the user and sent to the cloudlet goes through the steps shown in Fig. 1 and, therefore, is affected

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

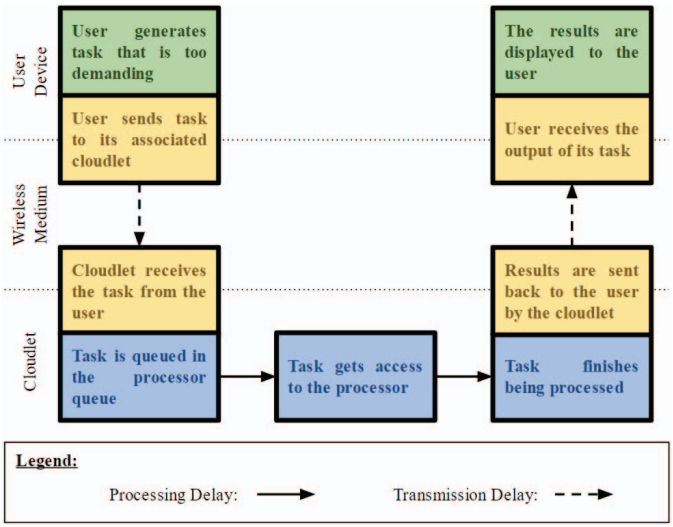

Fig. 1. Diagram of the Service Delay, where Transmission Delay and Processing Delay and clearly divided. Please note the steps and actions involved in each delay.

by Transmission Delay and Processing Delay. Approaches in the literature can be mainly divided into ones for improving Processing Delay and ones for improving Transmission Delay.

According to the literature, Processing Delay is actually minimized when the workload, i.e. the amount of associated users and corresponding VM servers, is equally divided among the cloudlets [8], [16]. This can be achieved by always associating users with the cloudlet currently hosting the least amount of VM servers; such cloudlet would be receiving fewer tasks, since it is associated with fewer users. The technique, called Work Scheduling [12], makes sure cloudlets with too much work do not receive new users, directing them to cloudlets with less work instead. For dealing with already connected users there is the technique VM Migration [8], where VM servers are migrated from overworked cloudlets to ones with less work. Both mentioned techniques aim at balancing the computation workload between cloudlets, making sure no cloudlet is overworked nor has wasted capacity.

The Transmission Delay, since it is composed of communication over the wireless medium, can be controlled by managing metrics related to the medium. For example, this can be done by controlling congestion through Access Point Scheduling [13], where users are associated with cloudlets hosting the fewest amount of VM servers; this guarantees that no cloudlet will have too many users connected to it (which would cause congestion and, consequently, delay). For controlling Signal to Interference plus Noise Ratio (SINR), there is Transmission Power Control [17], which configures the transmission power levels of the cloudlets, consequently allowing control over the Received Signal Strength (RSS) at the users and the channel capacity [18]. A more generic approach is Resource Allocation [19], [20], where resources are allocated to users as to provide a high QoS to all of them; the main point of this technique is guaranteeing fairness, even to users in less than favorable conditions.

All the references cited here deal with Service Delay by focusing only on Processing Delay or Transmission Delay. However, as discussed before, the ideal would be considering both of them together, since it would allow for a more effective control of Service Delay and a better QoS to a wider range of applications. There is one work which integrates both types of delay [11], but it does so in a pure mathematical format which, albeit possible for their considered scenario of only 2 cloudlets, is not feasible for larger cases. Our work is different because we deal with larger scenarios while continuing to be feasible. Therefore, in this paper we propose a method that mixes both VM Migration and Transmission Power Control to manage both Transmission Delay and Processing Delay in multiple cloudlets scenarios. Those techniques are mixed with a mathematical model of the Service Delay and a PSO model for achieving our goals. Moreover, while there are other delays involved, such as the Migration Delay spent on VM Migration, as said before, Transmission Delay and Processing Delay are the only relevant ones in our assumed scenario (under a different timescale, the other delays should be considered; for example, many users could lead to a high Migration Delay, which could be controled by avoiding many migrations if the Service Delay is already satisfactory).

III. Mathematical Model of Service Delay

Before presenting our proposal, we will first show a mathematical model for calculating the Service Delay in our assumed scenario. This model is based on an existing one in the literature [11] with adaptations to change it from an integral based model (which is good for optimizing, but complex for scenarios with many cloudlets) with a user density function, to one that considers the individual positions of the users.

A. User Association

We assume that each user associates with the cloudlet that produces the highest RSS value at the user. As mentioned before, this association results in the cloudlet hosting the VM server that corresponds to that user and, consequently, will handle the tasks created by that user. By this definition, we define the set of users associated to cloudlet n as

$$C_n = \{m \mid m \in M, S_n^m > S_i^m \ \forall i \in N, i \neq n\}, \quad (1)$$

where M is the set of all users, N is the set of all cloudlets, and S_q^w is the RSS between a sender q and a receiver w. Through this definition we can determine the amount of users associated with cloudlet n (which will use cloudlet n as access point) and the number of VM servers hosted by cloudlet n, respectively U_n and V_n.

$$U_n = |C_n|. \quad (2)$$

$$V_n = U_n. \quad (3)$$

Equation (3) comes from our assumption that each VM server only services a single user.

B. Processing Delay

Processing Delay depends on the amount of VM servers hosted in each cloudlet. We assume that the task arrival rate at each VM server follows a Poisson process with rate λ, which is the same for all users. From this, the total task arrival rate at cloudlet n is given by

$$\Lambda_n = V_n \cdot \lambda. \tag{4}$$

We assume the processors queue in each cloudlet follows a M/M/k queue model [21], with the required processing time of the tasks following an exponential distribution of rate $1/\mu$, and all cloudlets having k processors each. Thus, the average occupation rate per processor and the probability that a task has to wait in the queue in cloudlet n are

$$\rho_n = \frac{\Lambda_n}{k \cdot \mu}. \tag{5}$$

$$\Psi_n = \frac{(k \cdot \rho_n)^k}{k!} \cdot \left((1-\rho_n) \cdot \sum_{i=0}^{k-1} \frac{(k \cdot \rho_n)^i}{i!} + \frac{(k \cdot \rho_n)^k}{k!} \right)^{-1}. \tag{6}$$

From that, we have that the average time spent in the processor queue in cloudlet n is $\Psi_n \cdot (1-\rho_n)^{-1} \cdot (k \cdot \mu)^{-1}$. Thus, the average Processing Delay for clients of cloudlet n per task and the average Processing Delay per task across all users in the system are given respectively by

$$\hat{P}_n = \Psi_n \cdot (1-\rho_n)^{-1} \cdot (k \cdot \mu)^{-1} + \frac{1}{\mu}. \tag{7}$$

$$P_{\text{delay}} = \sum_{n \in N} (V_n \cdot \hat{P}_n) \Big/ \sum_{n \in N} V_n \ . \tag{8}$$

C. Transmission Delay

We assume that channel capacity follows the Shannon-Hartley Theorem [18], using SINR and Additive White Gaussian Noise spectral density [22]. The RSS can be estimated through the equation below:

$$S_x^y = 10^{(\omega_x + G_x + G_y + H - L_x^y)/10}/1000, \tag{9}$$

where ω_x is the transmission power of transmitter x, G_x and G_y are the total gains of x and y respectively, H is the Rayleigh power fading coefficient (for average, we use the value corresponding to 0.5 in the Rayleigh fading cumulative distribution function [23]), and L_x^y is the path loss between x and y, given by the Dual-Path Empirical Path Loss Model [24], all in decibels. The path loss model is below:

$$\begin{aligned} L_x^y = {} & L_1 + 10 \cdot n_1 \cdot \log_{10}(d(x,y)) \\ & + 10 \cdot (n_2 - n_1) \cdot \log_{10}\left(1 + \frac{d(x,y)}{d_b}\right). \end{aligned} \tag{10}$$

In (10), L_1 is the path loss at 1 meter of distance, n_1 and n_2 are the fading coefficients for respectively short and long distances, and d_b decides that classification. For each task, transmission is composed of the time needed to send the data plus the propagation time in the uplink and downlink. We will assume the user stays static, meaning the propagation time is the same in both directions. Thus, the average Transmission Delay per task in cloudlet n is

$$\hat{T}_n = \frac{2 \cdot t_n}{\gamma} + D_n^{\text{up}} + D_n^{\text{down}}, \tag{11}$$

where t_n is the average transmission distance between n and members of C_n, γ is the propagation speed, D_n^{up} is the average time to send in the uplink, and D_n^{down} is its downlink counterpart. For the last two values, propagation time is obviously excluded.

We assume round robin scheduling in the uplink [25]. Therefore, there is no interference between users of the same cloudlet, but there is interference between users of different cloudlets (only one user per cloudlet, since they follow a round robin scheme). The interference sensed at cloudlet n and the average time its associated users need to access the channel to send the task input are given by

$$I_n = \sum_{i \in N}^{i \neq n} \left(\sum_{j \in C_i} S_j^n \Big/ U_i \right). \tag{12}$$

$$Q_n^{\text{up}} = \frac{1}{U_n} \cdot \sum_{j \in C_n} \frac{p^{\text{up}}}{B^{\text{up}} \cdot \log_2\left(1 + \frac{S_j^n}{B_{\text{up}} \cdot \mathfrak{N} + I_n}\right)}, \tag{13}$$

where $\mathfrak{N}$ is the noise spectral density, p^{up} is average task input packet size, and B^{up} is the uplink channel bandwidth. Since we are using round robin scheduling, the access to the channel follows a M/D/1 queue [21], [25], where the timeslot of the round robin is constant and deterministic, and the arrival rate is a Poisson process with a rate determined by the queuing of new tasks and tasks who could not complete their transmission in a single timeslot returning to the end of the queue. The occupation rate of the channel to send to cloudlet n is given by the following equation from queuing theory:

$$\phi_n = \begin{cases} \lambda \cdot U_n + \frac{1}{\tau} \cdot \left(1 - \frac{\tau}{Q_n^{\text{up}}}\right) & \text{, if } Q_n^{\text{up}} < \tau \\ \lambda \cdot U_n & \text{, otherwise.} \end{cases} \tag{14}$$

In (14), τ is the length of the timeslot. The average time spent waiting in the queue for sending to cloudlet n is also given by queuing theory as

$$\varpi_n = \frac{\phi_n \cdot \tau}{2 \cdot (1 - \phi_n)}. \tag{15}$$

Each user must access the channel for as many timeslots as it is necessary to send the task input. Therefore, the average time for sending a packet in the uplink to cloudlet n is

$$D_n^{\text{up}} = \lceil Q_n^{\text{up}}/\tau \rceil \cdot (\varpi_n + \tau). \tag{16}$$

In the downlink, we utilize Orthogonal Frequency-Division Multiplexing [19]. This means each user has an individual

amount of allocated bandwidth, which we assume is proportional to the inverse of the logarithm base 2 of the RSS [11]. Thus, a user Z associated to cloudlet n will have allocated to itself the following amount of downlink bandwidth:

$$b_Z^{\text{down}} = B^{\text{down}} \cdot \frac{\left(\log_2\left(S_n^Z\right)\right)^{-1}}{\sum\limits_{j \in C_n}\left(\log_2\left(S_n^j\right)\right)^{-1}}, \tag{17}$$

where B^{down} is the total bandwidth in the downlink channel. Here, interference comes from the other cloudlets. Thus, the average time needed to send the output packet from cloudlet n to one of its associated users is

$$D_n^{\text{down}} = \frac{1}{U_n} \cdot \sum_{j \in C_n} \frac{p^{\text{down}}}{b_j^{\text{down}} \cdot \log_2\left(1 + \frac{S_n^j}{b_j^{\text{down}} \cdot \mathfrak{N} + \sum\limits_{i \in N} S_i^j}\right)}, \tag{18}$$

where p^{down} is the average task output packet size.

Finally, the average Transmission Delay per task across all users in the system is given by

$$T_{\text{delay}} = \sum_{n \in N}(U_n \cdot \hat{T}_n) \Bigg/ \sum_{n \in N} U_n \ . \tag{19}$$

D. Service Delay

The average Service Delay per task across all users in the system is

$$S_{\text{delay}} = P_{\text{delay}} + T_{\text{delay}}. \tag{20}$$

IV. Proposed Algorithm

Given the mathematical model of Service Delay from Section III, it is intuitive to mathematically optimize it by making the transmission power levels the decision variables (since they decide user association and RSS, they relate to both Processing Delay and Transmission Delay) and minimizing (20). This approach was taken in the literature before [11], using integrals and partial derivatives. The problem is that this is only feasible for small amounts of cloudlets (such as 2 in the reference), since that number corresponds to the number of transmission power levels and consequently decision variables. Moreover, brute force has a complexity that is exponential on the amount of decision variables, Linear Programming techniques do not apply since this is not a linear equation system, and derivatives and integrals only work with few decision variables (anything else is too complex).

This is why we utilize PSO for optimizing (20) instead. PSO [15] works with a set of particles that intelligently move in the search space (where each position is a possible solution), trying to improve the best local (i.e. for that particle) and global (i.e. among all particles) solutions. The intelligence part comes from the bias on movement, that tends to go towards the best solutions found so far. Through this mechanism, PSO is capable of nearly optimizing the fitness function at a low execution time [26]. For our PSO model, shown in Algorithm 1, the solutions will be configurations for the transmission power levels of all cloudlets; therefore, our search space is N-dimensional. Initial positions and speed for all particles are random. The fitness function, $f(\cdot)$, is Equation (20). R is the number of particles; L is the number of iterations; q_r, v_r and h_r are respectively the position, the speed and the best local solution of particle r; g is the best global solution; ϑ is the inertia constant; and ϱ_h and ϱ_g are the acceleration biases for the best personal and global solutions respectively.

Algorithm 1 PSO model for ECC multiple cloudlets scenario

for all $r \in R$ **do** initialize q_r as a random N-tuple
for all $r \in R$ **do** initialize v_r as a random N-tuple
while executed iterations $< L$ **do**
  for all $r \in R$ **do**
    $y \leftarrow f(q_r)$
    if $y < f(h_r)$ **then** $h_r \leftarrow q_r$
    if $y < f(g)$ **then** $g \leftarrow q_r$
    $\varphi_h \leftarrow randomInt(0, 1)$
    $\varphi_g \leftarrow randomInt(0, 1)$
    $v_r \leftarrow \vartheta \cdot v_r + \varphi_h \cdot \varrho_h \cdot (q_r - h_r) + \varphi_g \cdot \varrho_g \cdot (q_r - g)$
    $q_r \leftarrow q_r + v_r$
return g

Algorithm 2 Integrated transmission power and VM migration control for Service Delay minimization

1: Collect the physical location of users and cloudlets
2: Use Algorithm 1 to find configuration for lowering (20)
3: Set transmission power levels of cloudlets according to the configuration found
4: Decide user association based on Equation (1)
5: Execute VM migration if user association changed

In our proposal, a central office (CO), which aggregates information about users and cloudlets and can control the servers, would execute Algorithm 2. The CO would use as input the topography of the scenario, execute the PSO algorithm (which means solving the equation model from Section III) and, through this, find a transmission power configuration for all cloudlets that lowers the Service Delay. Fig. 2 illustrates this process. The frequency of execution of the procedure depends on the scenario; more dynamic cases should execute the algorithm more often, since the overhead would be compensated by the corrections to the Service Delay, while more static cases can execute the algorithm less often.

V. Comparative Study

To evaluate the performance of our proposal, we prepared three study cases, with parameters shown in Table I. Each study case was run 100 times, with different randomly generated topologies (i.e. physical location of users and cloudlets) each run. Results shown here are the average across all 100 runs, calculated through the model in Section III. We assume users are static, and bandwidth and average packet size are the same both in uplink and downlink. For the path loss model,

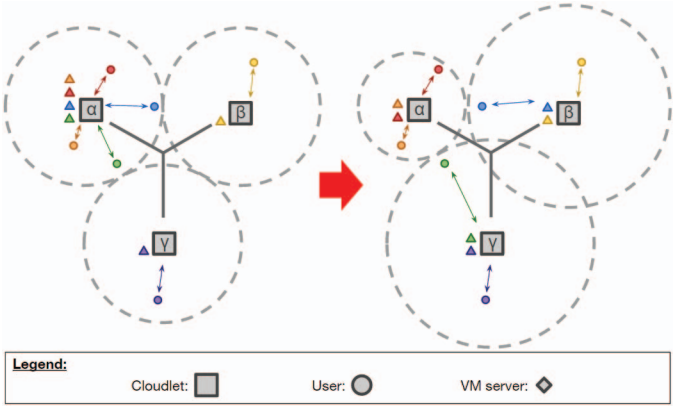

Fig. 2. Illustration of Procedure 2 being executed. In the initial moment, α is overworked while β and γ have wasted resources. After executing the procedure, the green and blue VM servers have been migrated away from α to balance the workload, and α has lowered its transmission power level while β and γ raised theirs, to better reflect their associated users' positions. This way, all cloudlets can efficiently reach their users, and work is equal.

TABLE I
STUDY CASES PARAMETERS

Study Case	1	2	3
Average packet size	500kB	500kB	0.5 to 1.5MB
Average task service time	500ms	50 to 1500ms	500ms
Number of cloudlets	3	40	
Number of users	120	500	
Number of PSO iterations	250	100	
Total area size	0.04km^2	0.25km^2	
Bandwidth (up and downlink)	1GHz		
User transmission power	27dBm		
User device total gain	8.35dBi		
Cloudlet total gain	24.5dBi		
Noise spectral density	4*10^{-19}W/Hz		
Wireless propagation speed	3*10^{8}m/s		
Processors per cloudlet	8		
Single user task arrival rate	6 tasks/min		
Round robin timeslot	85ms		

the level at 1m is 20dBm, the coefficients for short and long distances are respectively 2 and 4, and the breakpoint is 100m [24]. For the PSO model, there are 30 particles, the inertia constant is 0.7, and both accelerations are 2 [15].

For Study Case 1, we compared the performance of our proposal at each iteration with optimal values calculated through brute force. As seen in Fig. 3, the Proposed Approach reaches values close to optimality, with the difference being under 5ms after 50 iterations. However, the proposal manages this with an execution time 11.57% that of the method used for calculating the optimal value; and, as mentioned before, this value tends to be smaller in a higher scale. Thus, Study Case 1 shows how the Proposed Approach is a valid and still computationally feasible way of approaching optimality.

For Study Cases 2 and 3, we compared the Proposed Approach with two conventional ones. For the No Migration Approach, users send their tasks to the closest cloudlet, which also hosts their corresponding VM servers; this results in minimum transmission distance, but can lead to congestion, as too many users try to send packets to the same cloudlet, and overwork, as a single cloudlet hosts too many VM servers. In the Conventional Approach [8], users also send their tasks to the closest cloudlet, but now we assume VM servers are migrated as to result in all cloudlets hosting the same amount of VM servers; this gives minimum Processing Delay, but can still lead to congestion (tasks sent to cloudlets which do not host the corresponding VM server are transmitted through a wired connection of insignificant latency to the correct cloudlet, with the task output doing the opposite route afterwards). Regarding complexity, both conventional approaches and the proposal are $O(N \cdot M)$, where N and M are respectively the number of cloudlets and users. In Study Case 2, computation burden is varied in the form of the average task execution time, and in Study Case 3, communication burden is varied in the form of packet size. The results for Study Case 2 (Fig. 4) and Study Case 3 (Fig. 5) show how

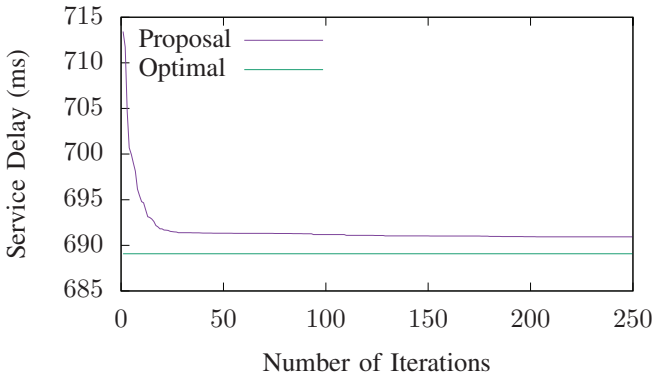

Fig. 3. Results for Study Case 1.

the No Migration Approach and the Conventional Approach have similar performances, while the proposal is consistently better. The difference to the Conventional Approach is smaller when computation use is high, since this approach minimizes Processing Delay, but the proposal has an advantage because it is the only one to consider Transmission Delay. This is more evident as the use of communication increases. The results show how a dual focus approach leads to significant improvement in performance when compared to single focus. It also shows how the proposal can deal with various application profiles (i.e. mix of computation and communication burdens) better than the conventional approaches. The improvement (of between 0.2s and 1.3s) may seem small for a single task, but users' jobs are composed of multiple tasks, increasing the importance of the proposal's performance enhancement.

VI. CONCLUSION

In this paper, we proposed a PSO-enhanced method of lowering Service Delay in ECC together with a mathematical model for calculating Service Delay. The proposed method,

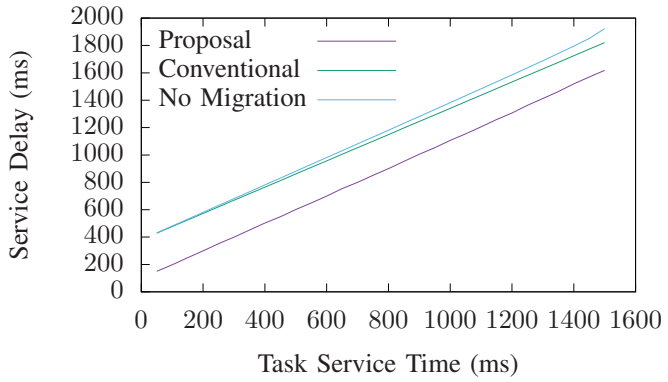

Fig. 4. Results for Study Case 2.

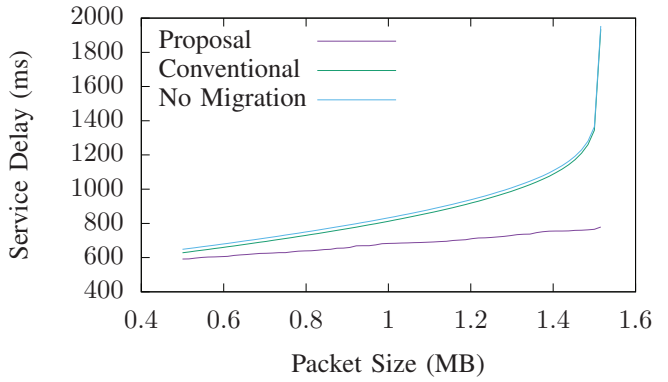

Fig. 5. Results for Study Case 3.

which considers both communication and computation elements, was shown to be significantly better than an existing approach which only improves Processing Delay. This corroborates our theory that a combined method of improving Transmission Delay and Processing Delay is the most efficient way of dealing with Service Delay. The proposal was also shown to be near optimal while being computationally feasible. Since the expectation in the next generation of mobile devices is to rely more on communication (due to higher data rates), our proposal will be even more relevant, since it was shown to be superior specially in scenarios with high transmission burdens. For future works, other types of delay (such as Migration Delay) and specific scenarios (such post disaster infrastructure setup) could be considered.

ACKNOWLEDGMENTS

This research is a part of Research and Development on Intellectual ICT System for Disaster Response and Recovery, the Commissioned Research of the National Institute of Information and Communications Technology (NICT), Japan.

REFERENCES

[1] D. Mercer. (2014) 33 Billion Internet Devices by 2020: Four Connected Devices for Every Person in the World. [Online]. Available: www4.strategyanalytics.com/default.aspx?mod=pressreleaseviewer&a0=5610

[2] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," in *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, May 1996, pp. 1–7.

[3] ——, "A Brief History of Cloud Offload: A Personal Journey from Odyssey Through Cyber Foraging to Cloudlets," *GetMobile: Mobile Comp. and Comm.*, vol. 18, no. 4, pp. 19–23, January 2015.

[4] H. Nishiyama *et al.*, "Relay by Smart Device: Innovative Communications for Efficient Information Sharing Among Vehicles and Pedestrians," *IEEE Vehicular Technology Magazine*, vol. 10, no. 4, pp. 54–62, December 2015.

[5] H. Nishiyama, M. Ito, and N. Kato, "Relay-by-Smartphone: Realizing Multihop Device-to-Device Communications," *IEEE Communications Magazine*, vol. 52, no. 4, pp. 56–65, April 2014.

[6] H. Chang *et al.*, "Bringing the cloud to the edge," in *Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2014, pp. 346–351.

[7] M. Satyanarayanan *et al.*, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, October 2009.

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

[9] G. Lewis *et al.*, "Tactical Cloudlets: Moving Cloud Computing to the Edge," in *Proceedings of the 2014 IEEE Military Communications Conference*, October 2014, pp. 1440–1446.

[10] W. Shi *et al.*, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, October 2016.

[11] T. G. Rodrigues *et al.*, "Hybrid Method for Minimizing Service Delay in Edge Cloud Computing through VM Migration and Transmission Power Control," *accepted at IEEE Transactions on Computers*, 2016.

[12] J. Oueis, E. C. Strinati, and S. Barbarossa, "The Fog Balancing: Load Distribution for Small Cell Cloud Computing," in *Proceedings of the 2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015, pp. 1–6.

[13] K. Suto *et al.*, "QoE-Guaranteed and Power-Efficient Network Operation for Cloud Radio Access Network With Power Over Fiber," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 127–136, December 2015.

[14] L. Yang *et al.*, "Cost Aware Service Placement and Load Dispatching in Mobile Cloud Systems," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1440–1452, May 2016.

[15] M. R. Bonyadi and Z. Michalewicz, "Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review," *Evolutionary Computation*, March 2016.

[16] A. Iosup *et al.*, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 931–945, June 2011.

[17] G. von Zengen *et al.*, "Transmission power control for interference minimization in WSNs," in *Proceedings of the 2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, August 2014, pp. 74–79.

[18] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[19] J. Armstrong, "OFDM for Optical Communications," *Journal of Lightwave Technology*, vol. 27, no. 3, pp. 189–204, February 2009.

[20] Z. M. Fadlullah *et al.*, "Cooperative QoS Control Scheme based on Scheduling Information in FiWi Access Network," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 2, pp. 375–3833, December 2013.

[21] I. Adan and J. Resing, *Queueing theory*. Eindhoven University of Technology Eindhoven, 2002.

[22] K. McClaning and T. Vito, *Radio Receiver Design*. Noble Publishing Corporation, 2000.

[23] B. Sklar, "Rayleigh fading channels in mobile digital communication systems. I. Characterization," *IEEE Communications Magazine*, vol. 35, no. 7, pp. 90–100, July 1997.

[24] L. W. Barclay, *Propagation of Radiowaves, 2nd Edition*. Institution of Engineering and Technology, 2003.

[25] G. Miao *et al.*, *Fundamentals of Mobile Data Networks*. Cambridge University Press, 2016.

[26] D. P. Tian, "A Review of Convergence Analysis of Particle Swarm Optimization," *International Journal of Grid and Distributed Computing*, vol. 6, no. 6, pp. 117–128, December 2013.