Availability Analysis of a Drone System with Proactive Offloading for Software Life-extension

Kengo Watanabe
Department of Computer Science
University of Tsukuba
Tsukuba, Japan
watanabe.kengo@sd.cs.tsukuba.ac.jp

Fumio Machida
Department of Computer Science
University of Tsukuba
Tsukuba, Japan
machida@cs.tsukuba.ac.jp

***Abstract*—Real-time image processing on a drone to recognize the real-world environment has become popular recently in many applications. However, continuous image processing on a drone may entail the degradation of performance and reliability over the long-time operation, also known as software aging. Since the degradation due to software aging progresses with the amount of the workload to process, offloading the image processing tasks to other computers can mitigate the progression of the software aging. In this paper, we propose a new software life-extension method to counteract software aging on a drone image processing system by means of proactive task offloading. To evaluate the effectiveness of the proposed method, we develop continuous-time Markov chains (CTMCs) to analyze the stochastic behaviors of the system. Through numerical experiments, we show that proactive offloading improves the steady-state availability, the mean time to down (MTTD), and the average throughput by 1.85%, 1.57x, 1.48x, respectively. We also show that the combination of offloading and software rejuvenating further improves the steady-state availability and the average throughput.**

***Keywords*—Drone, Fog computing, Software aging, Software life-extension, Offloading**

I. Introduction

As the performance and functionality of drones are evolving recently, drone-based systems are applied in various application domains. Modern intelligent drones can recognize their surroundings in real-time by processing images captured by cameras and dynamically decide the air route and actions for conducting their missions. Real-time adaptive behavior of such drones is especially required in mission-critical tasks such as disaster rescue and urban surveillance [1].

While continuous execution of image processing is essential for intelligent drone systems, a long-running high-workload process easily confronts the risk of software aging. Software aging is a phenomenon inducing performance degradation and an eventual system failure after a long-running operation due to aging-related software bugs [2]. Memory leak, memory fragmentation, and accumulated numerical errors are typical causes of software aging that may not be easily removed completely in the development phase [3]. Since the available computing resources (e.g., free memory) are extremely limited in a drone system, it is especially important to provide a dependable drone system with effective measures against the software aging problem.

The known conventional countermeasures to software aging include software rejuvenation and software life-extension. Software rejuvenation is a method to recover the system preventively before encountering failure due to software aging [2]. A typical method for software rejuvenation is a reboot that can clean up all the internal states in memory. However, frequent software rejuvenation can cause system availability degradation because the system is unavailable during software rejuvenation. On the other hand, software life-extension is a method to postpone the time to failure caused by software aging by controlling the amount of workload or resource affected by software aging [4][5]. Software life-extension is implemented in a virtualized system that allows dynamic additional resource allocation to a virtual machine suffering from software aging [4][5].

In this paper, we propose a new software life-extension technique to make a drone image processing system survive upon facing software aging due to the high workload of image processing. The proposed technique attempts to proactively offload computation tasks to other devices or servers so that the reduced workloads mitigate the progress of software aging on the process running on the drone. Unlike the conventional study, our software life-extension method does not necessitate a virtualization platform for fulfilling the resource. Instead, our approach uses a fog computing node to offload the computation tasks via a wireless communication link to the base station, which has been adopted in several recent studies of drone processing systems [6][7][8][9]. In order to evaluate the effectiveness of the proposed method, we construct continuous-time Markov chains (CTMCs) for representing the behaviors of systems without offloading, with offloading, and with offloading and rejuvenation. We conduct numerical experiments to evaluate the quantitative quality measures that are the steady-state availability, the mean time to down (MTTD), and the average throughput of image processing. The evaluation results show that our proactive offloading approach improves the steady-state availability, the MTTD, and the average throughput by 1.85%, 1.57x, and 1.48x, respectively, by delaying the occurrence of a failure due to software aging. When software rejuvenation is applied to the system in addition to proactive offloading, the steady-state availability can be further improved. We find that the steady-state availability is maximized at a specific combination of the mean time to offloading (MTTO) and the mean time to rejuvenation (MTTRJ). Furthermore, while the steady-state availability and the MTTD decrease as the MTTRJ becomes shorter, we also find that the average throughput can be improved by a short MTTRJ.

To summarize, we make the following contributions in this paper.

1) A new software life-extension method using proactive task offloading is proposed to counteract software aging in a drone processing system.
2) CTMC models are constructed for capturing the stochastic behaviors of the drone systems adopting proactive offloading and software rejuvenation.
3) Numerical experiments are conducted to show the effectiveness of the new software life-extension method with respect to the steady-state availability, the MTTD, and the throughput of image processing.

The rest of the paper is organized as follows. Section II describes the related work. Section III explains the configuration of the target system. Section IV presents the CTMCs for analyzing the steady-state availability, the MTTD, and the average throughput of drone systems. Section V introduces the quality measures for evaluation. Section VI shows the results of the numerical experiments to present the effectiveness of proactive offloading for software life-extension. Finally, Section VII presents the conclusion and briefly introduces the future works.

II. Related work

Computation offloading has been extensively studied recently in the edge computing context [10]. End devices often have limitations in terms of computation resources, and hence offloading computation tasks to an edge server is recognized as a viable solution. To determine where and how to offload computation tasks considering different aspects and design trade-offs, a variety of techniques such as machine learning, artificial intelligence, and control theory are applied [11]. Computation offloading in drone systems has been studied recently as well [6][7][8][9]. Although various offloading methods have been proposed, existing approaches are mostly reactive or adaptive to environmental conditions. In this paper, we focus on *proactive offloading*, in particular for mitigating the software aging experienced in end devices.

Software aging is a commonly-observed phenomenon in long-running software systems such as operating systems [12], web servers [13], and cloud computing infrastructure [14]. A recent experimental study reveals that image processing in an edge computing environment is also affected by software aging [15]. To the best of our knowledge, however, there is no existing study proposing a method to counteract software aging in a drone processing system.

While the techniques and models for software rejuvenation have been studied extensively in the literature [16], only a few works have been presented for software life-extension. Software life-extension was initially presented in [4], in which server virtualization is used to dynamically allocate additional resources to an aging server instance. This approach requires virtualization and may not be feasible in resource-constrained end devices. Therefore, this paper proposes a new method to extend the software life by offloading image processing tasks for a drone processing system.

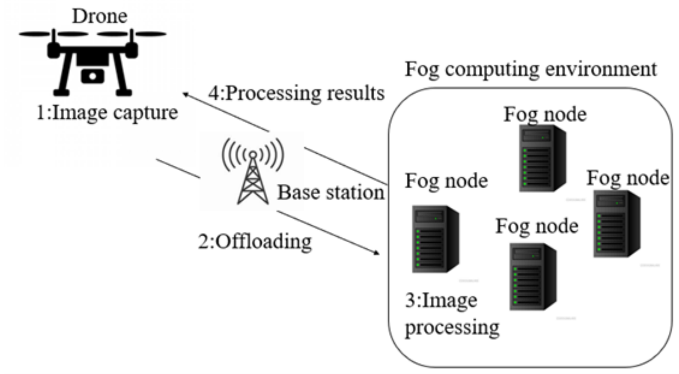


Fig. 1. System configuration

III. System configration

The configuration of the system considered in this paper is shown in Figure 1. The system consists of a drone that performs image processing during the flight and a fog computing environment that serves as an offload destination for image processing. The fog computing environment assigns a fog node to perform image processing when a drone requests offloading. For example, an edge computer or a virtual machine that can communicate via a wireless network such as Wi-Fi or 4G is assigned as a fog node. We assume software aging progresses by continuous operation of image processing as observed in [15]. When software aging is detected, the drone starts offloading the image processing tasks to a fog node. Offloading reduces the amount of computational processing on the drone and thereby lowers the failure rate of the process. If the process on the fog node fails during the offloading, the fog node process is aborted, and the process continues on the drone. If the process on the drone fails due to software aging, the recovery process is performed. In this study, we only consider process failures caused by software aging and do not consider other types of hardware or software failures.

IV. State transition model

This section details the CTMCs that represent the behaviors of a drone image processing system with software aging, life-extension and rejuvenation.

A. Aging model

The aging model shown in Figure 2 has 3 states. State 0 is the normal operating state in which the image processing on the drone is not affected by software aging. State 1 is the aging state in which the performance of image processing is deteriorated due to software aging. State 2 is the failure state in which the drone system cannot process any images due to

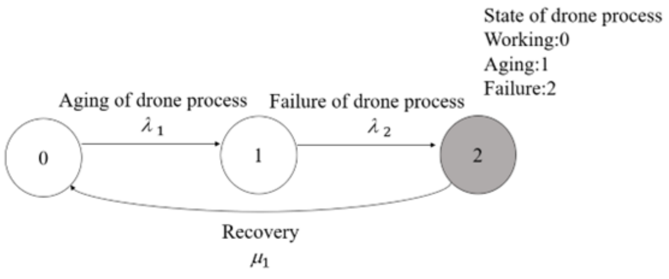


Fig. 2. Aging model

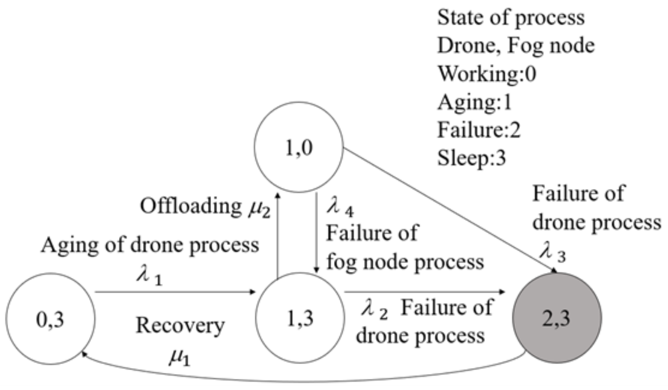


Fig. 3. Proactive offloading model

process failure. When software aging progresses on the drone, the system state is changed from the normal operating state to the aging state. We assume that the time to software aging follows the Exponential distribution with rate λ_1 as commonly assumed in the literature of software aging studies [17]. If the image processing process fails due to software aging, the state is changed to the failure state. We assume that the time to failure follows the Exponential distribution with rate λ_2 as assumed in the literature [17]. When the system encounters process failure, the process is recovered and returns back to the normal operation state. The recovery time is assumed to be exponentially distributed with rate μ_1 as commonly assumed in the literature of availability studies [18]. In this model, only state 2 is the down state which is represented by a shaded circle in Figure 2.

B. Proactive Offloading model

The proactive offloading model shown in Figure 3 extends the aging model by adding the state representing the offloading. State $(1,0)$ is the offloading state in which the image processing on the drone is offloaded to a fog node. When the drone offloads the image processing tasks to the fog node, the state is changed from the aging state to the offloading state. We assume that the MTTO follows the Exponential distribution with rate μ_2. If the image processing on the drone fails in the offloading state, the state is changed to the failure state. We assume that the failure time of the process in the offloading state is exponentially distributed with rate λ_3. If the image

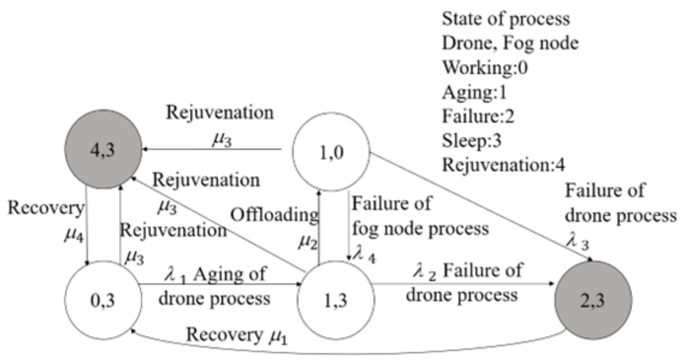


Fig. 4. Proactive offloading with rejuvenation model

processing task on the fog node fails due to software aging in the offloading state, offloading is aborted, and the system transitions to the aging state. We assume that the failure time of the fog node process follows the Exponential distribution with rate λ_4. In this model, state $(2,3)$ is the down state.

C. Proactive Offloading with Rejuvenation model

The proactive offloading with rejuvenation model shown in Figure 4 extends the proactive offloading model by adding the rejuvenation state. State $(4,3)$ is the rejuvenation state in which the image processing on the drone is rebooted. Software rejuvenation can be performed when the process on the drone is not failed, and when it does, the state is changed to the rejuvenation state. We assume that the MTTRJ follows the Exponential distribution with rate μ_3. In the rejuvenation state, the drone process is rebooted. When the rejuvenation completes, the process returns back to the normal operation state. The reboot time is assumed to be exponentially distributed with rate μ_4. In this model, state $(2,3)$ and state $(4,3)$ are the down states.

V. Analysis

To evaluate the reliability and performance of a drone image processing system subject to software aging, we introduce the quality measures, namely the steady-state availability, the MTTD, and the average throughput.

A. Steady-state Availability

An image processing service is available when the process is running either on the drone or the fog node. From the CTMCs, the steady-state availability can be computed by the sum of the probabilities that the system is in either one of the up states. Let $\boldsymbol{\pi} = (\pi_1, \pi_2, ..., \pi_n)$ be the steady-state probability vector and denote the set of up states of the model $k \in \{1,2,3\}$ as U_k, where 1, 2, and 3, represent the aging model, the proactive offloading model, and the proactive offloading with rejuvenation model, respectively. The steady-state availability of the system is given by

$$A_k = \textstyle\sum_{i \in U_k} \pi_i. \tag{1}$$

The transition rate matrices of the three models are given by

$$P_1 = \begin{pmatrix} 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_2 \\ \mu_1 & 0 & 0 \end{pmatrix}, \quad (2)$$

$$P_2 = \begin{pmatrix} 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & \lambda_2 & \mu_2 \\ \mu_1 & 0 & 0 & 0 \\ 0 & \lambda_4 & \lambda_3 & 0 \end{pmatrix}, \quad (3)$$

$$P_3 = \begin{pmatrix} 0 & \lambda_1 & 0 & 0 & \mu_3 \\ 0 & 0 & \lambda_2 & \mu_2 & \mu_3 \\ \mu_1 & 0 & 0 & 0 & 0 \\ 0 & \lambda_4 & \lambda_3 & 0 & \mu_3 \\ \mu_4 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (4)$$

The infinitesimal generator matrices Q_k is constructed from the transition rate matrix P_k. Then the steady-state probability $\boldsymbol{\pi}$ satisfies the following equation.

$$\boldsymbol{\pi} Q_k = 0. \quad (5)$$

Since the sum of the steady state probabilities is one, we have

$$\boldsymbol{\pi} \boldsymbol{e}^T = 1, \quad (6)$$

where $\boldsymbol{e}$ is an n-dimensional vector whose elements are 1. Solving equations (5) and (6) for $\boldsymbol{\pi}$ yields the steady-state probabilities. By equation (1) the steady-state availabilities for the aging model, the proactive offloading model, and the proactive offloading with rejuvenation model are derived as follows.

$$A_1 = \frac{1 + \frac{\lambda_1}{\lambda_2}}{1 + \frac{\lambda_1}{\lambda_2} + \frac{\lambda_1}{\mu_1}}, \quad (7)$$

$$A_2 = \frac{1 + \frac{(\lambda_3+\lambda_4+\mu_2)\lambda_1}{(\lambda_3+\lambda_4)(\lambda_2+\mu_2)-\lambda_4\mu_2}}{1 + \frac{\lambda_1}{\mu_1} + \frac{(\lambda_3+\lambda_4+\mu_2)\lambda_1}{(\lambda_3+\lambda_4)(\lambda_2+\mu_2)-\lambda_4\mu_2}}, \quad (8)$$

$$A_3 = \frac{X}{X+Y}, \quad (9)$$

$$X = (\mu_3 + \lambda_3 + \lambda_4)(\lambda_1 + \lambda_2 + \mu_2 + \mu_3) + \mu_2(\lambda_1 - \lambda_4),$$

$$Y = \frac{\lambda_1}{\mu_1}\{\lambda_2(\mu_3 + \lambda_3 + \lambda_4) + \lambda_3\mu_2\} + \frac{\mu_3}{\mu_4}X.$$

B. MTTD

The image processing service becomes unavailable when the process encounters failure or undergoes rejuvenation. The mean time to down (MTTD) from the time to start the process at the normal operating state can be expressed as the sum of the time spent in the up states before the system reaches either one of the down states. Denoting the average sojourn time vector as $\boldsymbol{t} = (t_1, t_2, ...t_n)$, the average time spent in each up state starting from the normal operating state can be expressed by

$$MTTD = \textstyle\sum_{i \in U_k} t_i. \quad (10)$$

TABLE I
PARAMETERS FOR NUMERICAL EXPERIMENTS

Variables	Values [1/hour]
Process recovery rate on a drone	$\mu_1 = 1$
Offloading rate	$\mu_2 = \frac{1}{MTTO}$
Software rejuvenation trigger rate	$\mu_3 = \frac{1}{MTTRJ}$
Process recovery rate in the rejuvenation state	$\mu_4 = 60$
Aging rate of a drone process	$\lambda_1 = \frac{1}{6}$
Failure rate of a drone process	$\lambda_2 = \frac{1}{12}$
Failure rate of a drone process in the offloading state	$\lambda_3 = \frac{1}{24}$
Failure rate of a fog node process	$\lambda_4 = \frac{1}{24}$
Service rate on a drone	$r_1 = 1200$
Service rate on a drone in the aging state	$r_2 = 600$
Service rate on a drone in the offloading state	$r_3 = 1200$

For the infinitesimal generator matrix Q_k, define Q_u as a submatrix consisting of only up states, and denote the corresponding subvector of $\boldsymbol{t}$ as $\boldsymbol{t}_u$. The average sojourn time satisfies the following equation [18].

$$\boldsymbol{t}_u Q_u = -(1, 0, ..., 0). \quad (11)$$

Solving equation (11) for $\boldsymbol{t}_u$ and using equation (10) yields the MTTD.

C. Average throughput

Besides the availability and reliability, the performance of the service also needs to be considered in system design. We consider the throughput of the image processing that quantifies the expected number of images processed per unit time. The average throughput of image processing can be computed from the steady-state probability in each state and the throughput of image processing at the state. Denote the processing rate in each state as $\boldsymbol{r} = (r_1, r_2, ...r_n)$, the average throughput can be given by

$$Average\,throughput = \textstyle\sum_{i \in U_k} \pi_i r_i. \quad (12)$$

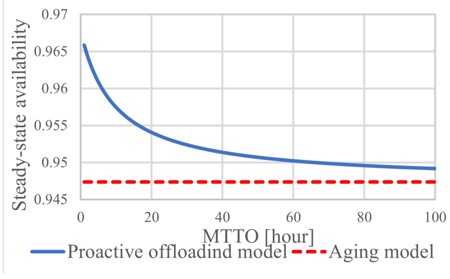
Note that the throughput is only counted in the up states, but the processing rate must decrease in the aging state.

VI. Numerical experiments

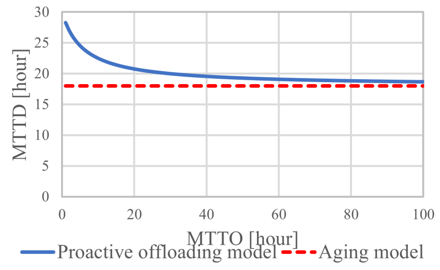
This section presents the results of numerical experiments to show the effectiveness of the proposed software life-extension technique.

A. Experimental plan

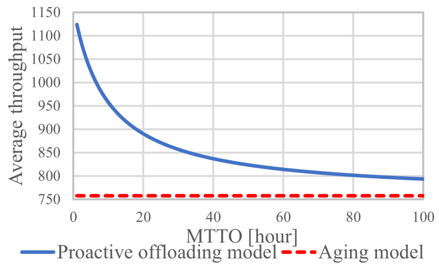
The parameter values used in the numerical experiments are shown in Table I. For the service rates and the failure rates, we set the values used in the previous studies [1][15]. For μ_2 and μ_3, which are reciprocal of MTTO and MTTRJ, we consider them as variables in the sensitivity analysis. First, to confirm the effectiveness of proactive offloading, the steady-state availability, the MTTD, and the average throughput are computed by varying the MTTO in the proactive offloading model. Next, to evaluate the effectiveness of proactive



(a) Steady-state availability

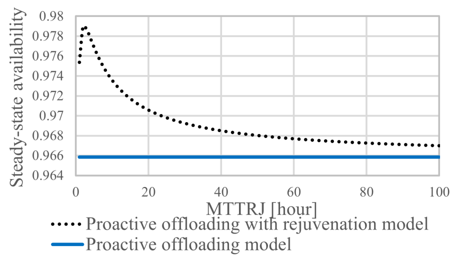


(b) Mean time to down (MTTD)

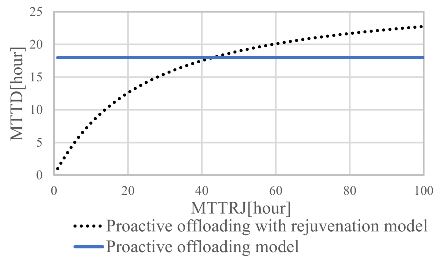


(c) Average throughput

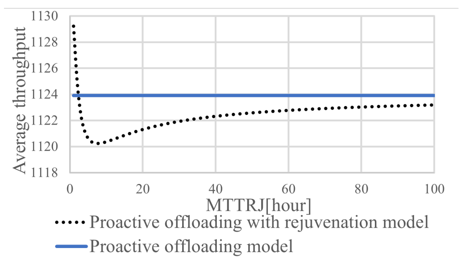
Fig. 5. Sensitivity analysis of the meant time to offloading (MTTO)



(a) Steady-state availability



(b) Mean time to down (MTTD)



(c) Average throughput

Fig. 6. Sensitivity analysis of the mean time to rejuvenation (MTTRJ)

offloading with rejuvenation, we compare the results of the proactive offloading model with rejuvenation model by varying the MTTRJ. Moreover, we investigate the optimal MTTRJ that maximizes the steady-state availability.

B. Results

1) Proactive Offloading model: The experiments are conducted by varying the MTTO from 1 hour to 100 hours at one-hour intervals in the proactive offloading model. The results of the steady-state availability, the MTTD, and the average throughput are shown in Figure 5(a), Figure 5(b), and Figure 5(c), respectively.

It can be seen that a shorter MTTO improves the steady-state availability, the MTTD, and the average throughput. Compared with the steady-state availability by the aging model, the value is improved to 0.965853659 (1.85%) by setting MTTO to one hour. Note that the availability can be further improved by setting MTTO as short as possible. In an extreme case, when $\mu_2 \rightarrow +\infty$, the steady-state availability reaches 0.967741935. The MTTD is improved to 28 hours, which is 1.57 times longer than the MTTD without using proactive offloading for software life-extension. In an extreme case, when $\mu_2 \rightarrow +\infty$, the MTTD value is increased up to 30. The average throughput is also improved to 1123.9, which is 1.48 times larger than the throughput without using proactive offloading. The average throughput can be maximized to 1161.3, by $\mu_2 \rightarrow +\infty$ in an extreme case.

2) Proactive offloading with rejuvenation model: Next, we fix the MTTO to an hour and vary the MTTRJ from 1 hour to 100 hours at one-hour intervals in the proactive offloading with rejuvenation model. The results of the steady-state availability, the MTTD, and average throughput are shown in Figure 6(a), Figure 6(b), and Figure 6(c), respectively.

From Figure 6(a), we observe that the steady-state availability is maximized at 2 hours of MTTRJ. As the MTTRJ increases over 2 hours, the availability decreases gradually and asymptotically approaches the steady-state availability of the proactive offloading model without rejuvenation. The results indicate that the system availability can be further improved by the combination of software life-extension and rejuvenation.

From Figure 6(b), we can observe that the MTTD is smaller than the MTTD in the proactive offloading model. The MTTD increases as the MTTRJ increases. This is reasonable as a shorter MTTRJ can easily make the system in down states by frequent rejuvenations.

On the other hand, as shown in Figure 6(c), the average throughput is higher than the average throughput in the proactive offloading model when MTTRJ is shorter than one hour. The throughput becomes smaller as the MTTRJ increases, and it asymptotically approaches the average throughput of the proactive offloading model. This is due to the fact that a shorter MTTRJ increases the probability of staying in the normal operating state which has a high processing rate.

In summary, while the steady-state availability of the system can be improved by proactive offloading with rejuvenation, we need to carefully determine the MTTRJ as it can potentially sacrifice the MTTD and the average throughput.

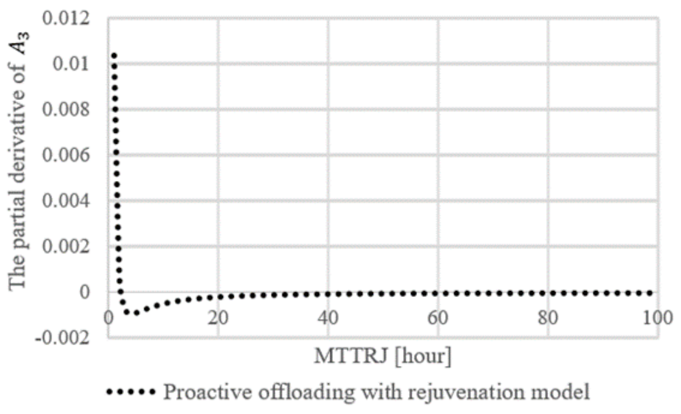


Fig. 7. The partial derivative of A_3 with respect to $\frac{1}{\mu_3}$

3) Optimal MTTRJ: As presented in Figure 6(a), the steady-state availability can be maximized at a certain MTTRJ. The optimal MTTRJ can be analyzed from (9). When we take the partial derivative of A_3 with respect to $\frac{1}{\mu_3}$, we have

$$A_3' = \frac{\partial A_3}{\partial \frac{1}{\mu_3}} = \frac{X'Y - XY'}{(X+Y)^2}, \tag{13}$$

$$X' = \frac{\partial X}{\partial \frac{1}{\mu_3}} = -{\mu_3}^2(\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \mu_2 + 2\mu_3),$$

$$Y' = \frac{\partial Y}{\partial \frac{1}{\mu_3}} = -\frac{\lambda_1}{\mu_1}(\lambda_2 {\mu_3}^2) + \frac{\mu_3}{\mu_4}(-\mu_3 X + X').$$

${A_3}'$ can be plotted as a function of MTTRJ ($= \frac{1}{\mu_3}$) as shown in Figure 7.

As the sign of ${A_3}'$ changes at 2.22378 from positive to negative as MTTRJ increases, 2.22378 [hours] is found to be the optimal MTTRJ that maximizes the steady-state availability in this system. Note that the optimal MTTRJ may change by choosing different values of MTTO. It is an interesting future work to explore the best combination of MTTO and MTTRJ considering the steady-state availability, the MTTD, and the average throughput.

VII. Conclusion

This paper presents proactive offloading as an effective countermeasure to software aging in a drone image processing system. To evaluate the effectiveness of the approach, we modeled the state transitions of the system using CTMCs and analyzed the models to evaluate the steady-state availability, the MTTD, and the average throughput. Numerical results show that the steady-state availability, the MTTD, and the average throughput are improved by proactive offloading. When software rejuvenation is used after proactive offloading, the steady-state availability can be further improved. We also show that the steady-state availability is maximized when we set MTTRJ to 2.22378 under our parameter configurations.

While we assumed an environment in which communication during offloading is always stable, the communication between the drone and the fog node may become unstable depending on the location, resulting in delays or failures in offloading. Future research will evaluate the effectiveness of the system, taking into account changes in the communication environment. It is also an important future work to validate the models with experimental studies.

Acknowledgment

This work is partly supported by the University of Tsukuba Research Foundation Support Program (S-type).

References

[1] F. Machida and E. Andrade, "PA-Offload: Performability-Aware Adaptive Fog Offloading for Drone Image Processing", 2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC), pp. 66-73, 2021.

[2] Y. Huang, C. Kintala, N. Kolettis and N. D. Fulton, "Software rejuvenation: analysis, modle and applications", Twenty-Fifth International Symposium on Fault-Tolerant Computing , pp. 381-390, 1995.

[3] M. Grottke, R. Matias and K. S. Trivedi, "The fundamentals of software aging", 2008 IEEE International Conference on Software Reliability Engineering Workshops (ISSRE Wksp), pp. 1-6, 2008.

[4] F. Machida, J. Xiang, K. Tadano and Y. Maeno, "Software Life-Extension: A New Countermeasure to Software Aging", 2012 IEEE 23rd International Symposium on Software Reliability Engineering, pp. 131-140, 2012.

[5] F. Machida, J. Xiang, K. Tadano and Y. Maeno, "Lifetime extension of software execution subject to aging", IEEE Transactions on Reliability, vol. 66, no. 1, pp. 123-134, 2017.

[6] J. Chen, S. Chen, S. Luo, Q. Wang, B. Cao, and X. Li, "An intelligent task offloading algorithm (iTOA) for UAV edge computing network", Digital Communications and Networks, vol. 6, no. 4, pp. 433-443, 2020.

[7] J. Yao and N. Ansari,"Online task allocation and flying control in fog-aided internet of drones", IEEE Transactions on Vehicular Technology, vol. 69, no. 5, pp. 5562-5569, 2020.

[8] J. Yu, A. Vandanapu, C. Qu, S. Wang, and P. Calyam, "Energy-aware dynamic computation offloading for video analytics in multi-UAV systems", in 2020 International Conference on Computing, Networking and Communications (ICNC), pp. 641-647, 2020.

[9] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, and M. Satyanarayanan, "Bandwidth-efficient live video analytics for drones via edge computing", in IEEE/ACM Symposium on Edge Computing (SEC), pp. 159-173, 2018.

[10] H. Lin, S. Zeadally, Z. Chen, H. Labiod, L. Wang, "A survey on computation offloading modeling for edge computing", Journal of Network and Computer Applications, vol. 169, 2020.

[11] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective", Computer Networks, vol. 182, 2020.

[12] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, "Software aging analysis of the Linux operating system", In IEEE International Symposium on Software Reliability Engineering (ISSRE), pp. 71-80, 2010.

[13] M. Grottke, L. Li, K. Vaidyanathan, and K. S. Trivedi, "Analysis of software aging in a web server", IEEE Transactions on reliability, vol. 55, no. 3, pp. 411-420, 2006.

[14] R. Pietrantuono and S. Russo, "A survey on software aging and rejuvenation in the cloud", Software Quality Journal, vol. 28, pp. 7-38, 2020.

[15] E. Andrade, F. Machida, R. Pietrantuono, D. Cotroneo, "Software Aging in Image Classification Systems on Cloud and Edge," 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 342-348, 2020.

[16] D. Cotroneo, R. Natella, R. Pietrantuono, S. Russo. 2014. "A survey of software aging and rejuvenation studies." J. Emerg. Technol. Comput. Syst. vol. 10, no. 1, pp. 1-34, 2014.

[17] T. Dohi, K. Trivedi, A.Avritzer "Handbook of Software Aging and Rejuvenation Fundamentals, Methods, Applications, and Future Directions." World scientific, 2020.

[18] K. Trivedi, A. Bobbio, "Reliability and Availability Engineering," Cambridge University Press, 2017.