PA-Offload: Performability-Aware Adaptive Fog Offloading for Drone Image Processing

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

Ermeson Andrade
Department of Computing
Federal Rural University of Pernambuco
Recife, Brazil
ermeson.andrade@ufrpe.br

***Abstract*— Smart drone systems have built-in computing resources for processing real-world images captured by cameras to recognize their surroundings. Due to limited resources and battery constraints, resource-intensive image processing tasks cannot always run on drones. Thus, offloading computation tasks to any available node in a fog computing infrastructure can be considered as a promising solution. An important challenge when applying fog offloading is deciding when to start or stop offloading, taking into account performance and availability impacts under varying workloads and communication link states. In this paper, we present a performability-aware adaptive offloading scheme called PA-Offload that controls the offloading of image processing tasks from a drone to a fog node. To incorporate uncertainty factors, we introduce Stochastic Reward Nets (SRNs) to model the entire system behavior and compute a performability metric that is a composite measure of service throughput and system availability. The estimated performability value is then used to determine when to start or stop the offloading in order to make a better trade-off between performance and availability. Our numerical experiments show the effectiveness of PA-offload in terms of performability compared to non-adaptive fog offloading schemes.**

Keywords—drone, fog computing, image processing, offload, stochastic model

I. INTRODUCTION

Recently applications of drone systems are prevailing in many domains as the performance and intelligence of these systems are improving. A smart drone system can recognize the surroundings by processing images captured by cameras and decide appropriate actions under given conditions. A real-time adaptive behavior of a drone is especially required in mission critical tasks such as disaster rescue and urban surveillance [1][2]. While quick and reliable responses are necessary in emergencies, image processing tasks are often too expensive to continuously run on the drone which has limited resources and short battery life. Thus, alternative processing modes may help improving the performance of smart drone systems.

Offloading image processing tasks to a fog computing infrastructure can be considered as an alternative processing mode that can complement the processing of drones [3]. The devices comprising the fog infrastructure are known as fog nodes, and they can be, for instance, a smart device connected to the drone, a small local computer, or any accessible virtual machines. Since drone's workload can be reduced by offloading and processing on fog nodes, which offer relatively stable execution environments, we can expect higher system availability (i.e., high probability the system is ready for image processing). On the other hand, the performance of image processing may degrade due to additional communication delay to send images to a fog node. The availability and performance can also be affected by uncertainty factors such as changes in workload intensity and communication link quality. It is not a trivial issue to decide when to start offloading considering the performance-availability trade-offs under varying workloads and communication link quality.

In this paper, we propose a novel adaptive offloading decision scheme called *PA-Offload* that determines the condition to start or stop the fog offloading based on the workload and the quality condition of the wireless communication links. To consider the offloading impacts on the system performance and availability, we introduce a performability measure [6] that is defined as a composite measure of service throughput and system availability in our application context. The expected performability measure is used to decide when to offload image processing tasks to a fog node considering the workload state and the communication link status. In order to quantitatively analyze the performability, we model the system behavior with Stochastic Reward Nets (SRNs) [5]. SRNs are variant of stochastic Petri nets that has the capability to assign rewards to individual states of the system, and hence allowing to compute the defined performability measure. We construct two models corresponding to the drone processing mode and the fog offloading mode, respectively. The performability measure can be computed through the solution of the SRNs. We conduct numerical experiments to show the effectiveness of PA-Offload. When compared to the drone processing mode and fog offloading mode, PA-Offload always achieves the highest performability under our experimental configuration. Our results also reveal that PA-Offload can effectively switch the mode in response to the changes in the workload state and the communication link status.

To sum up, the main contributions of this work are as follows.

1. We propose PA-Offload as a performability-aware adaptive fog offloading decision scheme for image processing tasks running on drones. PA-Offload enables to adapt to changes in workload intensities and quality of communication links.
2. We present comprehensive analytical models using SRNs for analyzing performance-availability trade-offs in image processing tasks running on drones and fog nodes. The solution of SRNs allows us to compute service throughput and system availability.

3. Through the numerical experiments, we confirm the effectiveness of PA-Offload that can achieve a better performability by adapting to the changes of workload states and communication link quality. Considering our parameter settings, PA-Offload outperforms the drone processing mode and fog offloading mode in terms of performability.

The rest of the paper is organized as follows. Section II describes the related work. In Section III, we clarify our problem scope regarding the fog offloading decision. In Section IV, we detail our approach called PA-Offload. Section V presents the SRNs for analyzing the performance and availability of drone image processing systems. Section VI shows the results of the numerical experiments to present the effectiveness of PA-Offload. Finally, Section VII presents the conclusion and briefly introduces the future works.

II. Related work

Offloading techniques have been investigated for fog computing as well as mobile cloud and mobile edge computing. By offloading the computation tasks, mobile or edge devices can reduce the workloads, and hence, extend their battery lives. A crucial aspect of offloading is to decide whether to offload or not, and what should be offloaded [11]. Offloading techniques can be broadly classified into full offloading or partial offloading [12]. In our study, we focus on the offloading of image processing tasks that is a part of an application program. Therefore, we consider a partial offloading, but the process to be offloaded is not changed. Many existing studies for full or partial offloading proposed methods to minimize the execution delay [13][14][15], to minimize the energy consumption [16][17], or to find a proper trade-off between the energy consumption and the execution time [18][19]. However, to the best of our knowledge, PF-offload is the first offloading decision scheme that can make a better trade-off between the system availability and the service performance.

Despite the growing attention to drone applications using cloud or fog computing resources, only a few studies addressed fog offloading issues. In [21], the authors presented a Fog-aided Internet of Drones (IoD) networks that employ fog nodes to provide computing resources to tasks offloaded from drones. The authors of [22] considered offloading decisions for real-time video analytics on drones to meet users' Quality of Experience (QoE) expectations. When multiple drones are used for a collaborative task, swarm of drones can be regarded as a part of fog computing [20]. While existing studies consider the application performance and cost factors in the task allocation or offloading decision, none of them addressed the reliability and availability trade-offs.

Performability modeling and analysis have been applied in many applications to compute performance-related metrics of degradable systems. Markov reward model (MRM) is introduced for analyzing the performability of a multi-processor system where both the system unavailability and the throughput loss are considered [23]. MRM is also used for analyzing the performability of a wireless network communication system subject to communication channel failures [24]. The performability of storage systems configured with Redundant Array of Independent Disks (RAID) is analyzed by Markov regenerative process in [25]. In fog computing studies, while several studies considered reliability and availability aspects, only a few studies addressed the performability attributes [26][27]. Our study is the first attempt to quantify the performability of a fog-assisted drone image processing system for effectively decide when to start or stop the offloading.

III. Problem scope

This section first introduces our target smart drone system. Next, we consider an offloading method using fog computing. Then, we discuss the issue of offload decision.

A. Target system: smart drone systems

Modern drones are equipped with cameras and high-speed processors to analyze real-time images. Local image processing in real time allows the situation awareness of drones such as collision avoidance and detection of objects (e.g., people or animals). Considering the use of a smart drone for a disaster rescue scenario, such as finding disaster victims in a devastated area, the situation awareness is essential [1]. In such a scenario, the operator may not directly control the drone due to geographical constraints, and hence the autonomous flight control is necessary particularly when the wireless communication links are not stable. Recognizing the surroundings of a drone through real-time image processing is vital to mission-critical drone application systems. In this work, we assume that the image processing application uses deep learning models to detect obstacles or objects in the captured images, but the proposed SRNs are agnostic to a specific type of image processing.

B. Offloading drone's image processing tasks

Since image processing is a resource-intensive and battery-consuming task, it is not always efficient to run the image processing on the drone that has limited computing resources and short battery life. Offloading tasks to fog nodes can be considered as an alternative computation mode for drone's image processing [3]. Fog computing is a horizontal system-level architecture that distributes computing resources close to the users [4]. No matter what type of fog node is assigned for the task (e.g., a local computer or a remote virtual machine), the drone can benefit from offloading by saving the battery and reducing the risk of a system failure associated with workload intensities. On the other hand, it may involve additional overheads on the performance of real-time image processing tasks because of the communication delay between the drone and the fog node.

C. Challenges in offloading decision

There are performance-availability trade-offs in choosing a drone processing mode or a fog offloading mode. In general, the drone processing mode is preferable in terms of real-time performance, while the fog computing mode is better in terms of system availability because the processing on the fog node can be more stable. However, the performance and availability of a system are significantly affected by decision-independent (exogenous) uncertainties from a real-world environment. One important environmental factor is the workload intensity of the image processing application. The frequency with which images are processed must depend on the mission of the drone system. For example, a drone system used for a disaster rescue requires higher rate of image processing than usual when the drone finds

or approaches a victim. Considering the system availability, high workload of image processing requests may be better to be shipped to the fog node if any fog node is available and accessible.

Another critical environmental factor is the quality of the wireless communication link. The drone can communicate with other terminals or a fog node via mobile or Wi-Fi networks. The quality of the communication link can change during the mission, or even worse the drone may lose the connection temporarily. When the communication quality significantly degrades, offloading the image processing is not a good decision in terms of both performance and availability perspectives.

It is not a trivial issue to decide when to start or stop the fog offloading in order to make a better trade-off between performance and availability under such environmental uncertainties. The system needs to continuously monitor the environmental status and estimate the performance and availability of the system, so that it can decide when to start or stop the fog offloading. In this paper, we address this problem and propose an approach based on stochastic models to dynamically change the processing mode according to the environmental states.

IV. PERFORMABILITY-AWARE ADAPTIVE FOG OFFLOADING

In this section, we present the *PA-offload* as an offloading decision scheme for drone image processing systems with higher performance and availability. To achieve a better balance between the performance and availability, we introduce a performability measure as a decision criterion for the fog offloading that is explained in Section IV-B.

A. System architecture

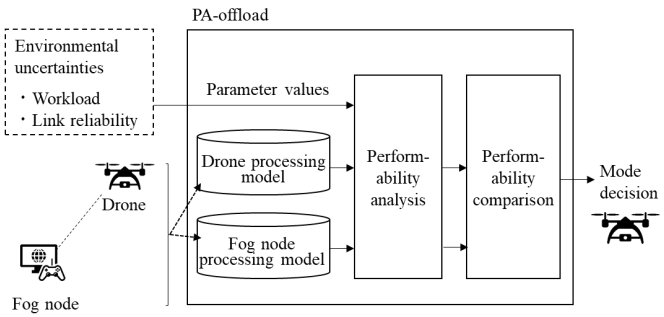


Figure 1. Architecture overview of PA-offload

Figure 1 shows the conceptual flow of the PA-offload. In response to input information on environmental uncertainties, PA-offload conducts performability analysis to decide the computation mode; either drone processing or fog node processing. From the system configuration (including a drone, a client or a fog node and the network link), we first construct the analytic model to represent the system behavior of the drone processing mode and fog offloading mode. The constructed models are deployed on PA-offload, so that the framework can dynamically compute the estimated performance measure with the input parameter values. Note that for the input parameter values, PA-offload takes into account two types of uncertainties: the workload intensity and the link reliability. The workload intensity is given by the arrival rate of image processing requests, while the link reliability is given by the communication disconnection rate. With this information, performability analysis module computes the expected performance and availability measures for individual processing modes. Subsequently, the performability comparison module computes the estimated performability measure for deciding whether to start or stop the offloading.

B. Performability computation

The key to decide whether to change the mode or not in PA-Offload is the estimation of performability. Performability is a composite performance-availability measure to evaluate the effectiveness of degradable systems [6]. Many practical application systems require not only high-performance but also need high-availability during their operation. A high-performance but unavailable system may not meet users' expectations, while a highly available system with low performance is not acceptable either. Performability can effectively incorporate these two factors in a unified measure, so that it is used for designing efficient system configurations or control policies.

For a drone image processing system, we are interested in the request processing capability during its operation period. The average request processing rate over the available period can be considered as the concerned performability measure. Let s be a state of a drone system and denote $p(s)$ as the performance measure when the system is in state s. The performability measure can be defined as

$$P_a = \int_{s \in A} p(s) dF(s),$$

where A represents the set of states where the system is available and $F(s)$ is the probability distribution of the random variable for the state s. To compute the value of P_a, the expected performance for each state s, as well as the probability of the state s need to be estimated. These values are not easily given from the system specification, since state changes are induced by environmental uncertainties. Therefore, we introduce stochastic models to capture state transitions of the system under environmental uncertainties. Details of the model are presented in Section V.

C. Mode decision

The output of PA-offload is the recommendation of the computation mode, which is expected to achieve a better performability. When PA-offload receives a set of parameter values $\boldsymbol{\theta}$ representing the states of the workload and communication link, the performability analysis module computes the expected performance and availability measures for drone processing mode and fog offloading mode simultaneously. Then, the performability comparison module evaluates the expected performability $E[P_{a,D}|\boldsymbol{\theta}]$ and $E[P_{a,F}|\boldsymbol{\theta}]$ for drone and fog offloading modes, respectively. The preferable mode $\mathcal{M}_\theta$ is determined by

$$\mathcal{M}_\theta = \underset{i \in \{D,F\}}{\operatorname{argmax}} E[P_{a,i}|\boldsymbol{\theta}].$$

If the current computation mode is the drone processing and the recommendation is F (fog node processing), then it is encouraged to start the fog offloading. On the other hand, if the

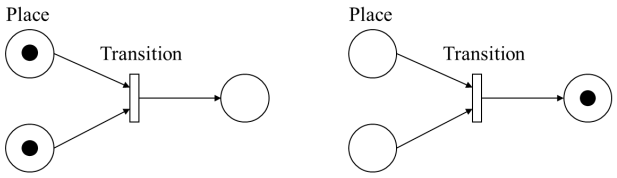
current mode is the fog processing mode and the recommendation is *D* (drone processing), then the system is requested to stop the offloading. In order to make a further improved control, we should also consider the delay and the cost incurred by mode changes, which is not considered in this paper and is regarded as a future work.

V. PERFORMABILITY MODEL

This section details the SRNs for evaluating the performability of a drone image processing system. Frist, we briefly introduce the formalism of SRNs.

A. Introduction to SRN

SRN is a variant of Stochastic Petri Nets (SPN) that has been extensively used in studies of availability and performance analysis, so that it provides a higher-level graphical representation of stochastic behavior of systems [5]. A Petri net is essentially represented by a bipartite directed graph with two types of nodes: places and transitions. A marking consists of tokens in the places of a Petri net that represents a particular system state. State transitions are represented by the change of marking in accordance with the transition firing rule. A transition can fire when all input places have a required number of tokens. When a transition fires, the tokens in all input places are removed and new tokens are placed in output places (see an example shown in Figure 2).



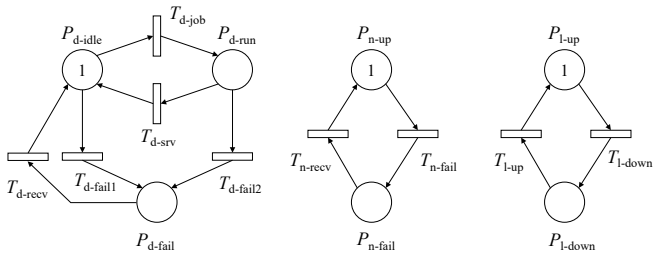
(a) Marking before the transition fires (b) Marking after the transition fires

SRN supports two types of transition, i.e., timed transition and immediate transition and two types of functions, i.e., guard and reward functions. A timed transition has a firing delay when it is enabled, while an immediate transition has zero firing time. They are represented by a rectangle and a thin black bar, respectively. Guard functions can specify the condition to enable a transition. To compute various performance measures, it is essential to specify reward functions. A reward function assigns rewards to each tangible marking of the Petri net.

Software packages such as SPNP [7] and SHARPE [8] support the solution of SRNs. When transition times are exponentially distributed, the SRNs are transformed into the equivalent continuous-time Markov chain (CTMC) with rewards. The expected reward rates defined by reward functions are computed through the steady-state solution of the CTMC. For efficient analysis in PA-Offload, in this paper, we assume that all timed transitions have exponentially distributed firing times. More details about SRN formalism, solution techniques and modeling examples can be found in the literature [5][10].

B. Drone processing model

We compose the SRN for the drone image processing system without using fog offloading. Figure 3 shows the SRN that consists of three inter-dependent subnets corresponding to a drone, a client node, and the wireless link between them.



(a) Drone model (b) Client node model (c) Wireless link model

The drone model represents the states of the job processing on the drone. When a token is deposited in $P_{\text{d-idle}}$, representing that the drone has no image for processing, the transitions $T_{\text{d\_job}}$ and $T_{\text{d-fail}}$ are enabled. If $T_{\text{d-job}}$ fires first, a token is removed from $P_{\text{d-idle}}$ and a new token is deposited in $P_{\text{d-run}}$. The state transition corresponds to the arrival of an image processing request. We assume that the arrival of job processing requests follows Poisson process with rate γ. The firing of $T_{\text{d-srv}}$ represents the completion of a job processing. We assign service rate ν_d for $T_{\text{d-srv}}$. On the other hand, if $T_{\text{d-fail}}$ fires first, which represents a drone process failure event from the idle state, a new token is deposited in $P_{\text{d-fail}}$. We assign the process failure rate λ_d for $T_{\text{d-fail1}}$. The drone process can fail also during image processing through the firing of the transition $T_{\text{d-fail2}}$. The process failure rate λ_{d2} is assigned for $T_{\text{d-fail2}}$, which is assumed to be larger than λ_d because the probability of a process failure increases during the execution. When a token is deposited in $P_{\text{d-fail}}$, the transition $T_{\text{d-recv}}$ is enabled, which represents the recovery of the process. We assign the process recovery rate μ_d for $T_{\text{d-recv}}$.

The client node model captures the failure-recovery behavior of a client device that communicates with the drone to receive the results of the image processing (e.g., alert message). Although the availability of a client node does not affect the image processing process on a drone, the user may not access the results of the process when the node is down. Thus, we need to consider the state of the client node to compute the system availability. A token deposited in $P_{\text{n-up}}$ is removed when $T_{\text{n-fail}}$ fires, representing a node failure event. Meanwhile, a token deposited in $P_{\text{n-fail}}$ is removed when $T_{\text{n-recv}}$ fires, representing a node recovery event. We assign the client failure rate λ_n and the client recovery rate μ_n to $T_{\text{n-fail}}$ and $T_{\text{n-recv}}$, respectively.

The wireless link model captures the states of the communication link between the drone and the client node. Even when both the drone and client node run properly, disconnection of the wireless communication link leads to the unavailability of the system. When a token is deposited in $P_{\text{l-up}}$, it represents the link is available and, at the same time, the transition $T_{\text{l-down}}$ is enabled. The firing of $T_{\text{l-down}}$ corresponds to a disconnection of the communication link. While a token is deposited in $P_{\text{l-down}}$, the transition $T_{\text{l-up}}$ is enabled. The firing of $T_{\text{l-up}}$ represents the reconnection of the link. The rates of link failure and reconnection are assumed to be λ_l and μ_l, respectively.

We define three reward functions for computing system availability and performance measures.

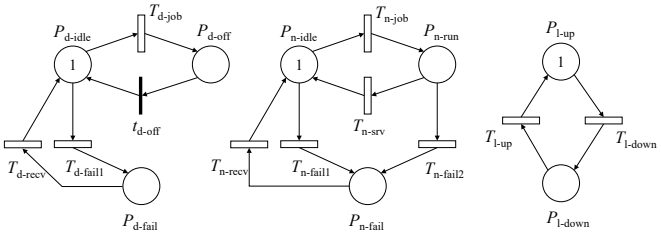
TABLE I. REWARD FUNCTIONS FOR DRONE PROCESSING MODE

Name	Measure	Function
svavail	Service availability	if ((#$P_{\text{d-idle}}$==1 or #$P_{\text{d-run}}$==1) and (#$P_{\text{n-up}}$==1) and (#$P_{\text{l-up}}$==1)) then 1 else 0
sysavail	System availability	if ((#$P_{\text{d-idle}}$==1 or #$P_{\text{d-run}}$==1) and (#$P_{\text{n-up}}$==1)) then 1 else 0
svthru	Service throughput	prob(#$P_{\text{d-run}}$==1) * ν_d

TABLE I shows the definition of the reward functions. For computing service availability, *svavail* assigns one reward for the marking (#$P_{\text{d-idle}}$==1 or #$P_{\text{d-run}}$==1) and (#$P_{\text{n-up}}$==1) and (#$P_{\text{l-up}}$==1), where #P_{x} represents the number of tokens in P_{x}. The specified condition represents that the service is available only when the drone, the client node and the wireless link are available simultaneously. We also define *sysavail* that exclude the condition on the communication link state to compute the system availability (the availability of the drone and the client). On the other hand, for job processing performance, *svthru* is defined as the effective throughput of $T_{\text{d-srv}}$ that can be computed by multiplying the probability of marking #$P_{\text{d-run}}$==1 and ν_d.

C. Fog offloading model

Next, we compose the SRN for the drone image processing system using the fog offloading mode. Figure 4 shows the SRN that consists of three inter-dependent subnets corresponding to a drone, a fog node and the wireless link between them. While the wireless link model is unchanged, the drone model and fog node model are changed from the drone processing models shown in Figure 3. TABLE II shows the guard functions assigned to the transitions of the subnets.



(a) Drone model (b) Fog node model (c) Wireless link model

Figure 4. SRN for the fog offloading mode

TABLE II. GUARD FUNCTIONS FOR THE FOG NODE PROCESSING MODE

Name	Transition	Function
goffload	$T_{\text{d-job}}$	if ((#$P_{\text{n-idle}}$==1) and (#$P_{\text{l-up}}$==1)) then 1 else 0
gfogjob	$T_{\text{n-job}}$	if (#$P_{\text{d-off}}$==1) then 1 else 0
gjobstart	$t_{\text{d-off}}$	if (#$P_{\text{n-idle}}$==0) then 1 else 0

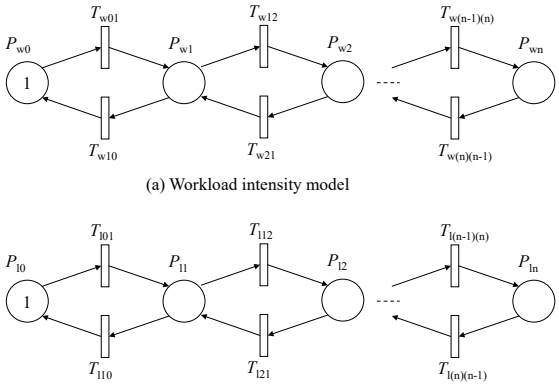
In the fog offloading mode, the captured images are not processed on the drone, but sent to the fog node through the network. Therefore, the drone model has the place $P_{\text{d-off}}$, representing fog offloading, instead of $P_{\text{d-run}}$. When a token is deposited in $P_{\text{d-off}}$ after firing $T_{\text{d-job}}$, the transition $T_{\text{n-job}}$ in the fog node model is enabled. The enabling condition is specified in the guard function *gfogjob* as shown in TABLE II. When $T_{\text{n-job}}$ fires, a token in $P_{\text{n-idle}}$ is removed and a new token is deposited in $P_{\text{n-run}}$. Simultaneously, the immediate transition $t_{\text{d-off}}$ fires by the guard function *gjobstart*. We assume that the average time to start a job processing in the fog node (communication delay) is $1/\omega$. During the fog node processing, the process may finish the job or fail, which are represented by the transitions $T_{\text{n-srv}}$ or $T_{\text{n-fail}}$, respectively. When $T_{\text{n-srv}}$ fires first, where the service rate is given by ν_n, a new token is deposited in $P_{\text{n-idle}}$. On the other hand, if $T_{\text{n-fail}}$ fires first, where the failure rate is given by λ_{n2}, a new token is deposited in $P_{\text{n-fail}}$. The recovery process is the same as in the previous model. Since offloading only works when the fog node is idle and the wireless communication link is available, such condition is specified by the guard function *goffload* assigned to $T_{\text{d-job}}$.

TABLE III. REWARD FUNCTIONS FOR THE FOG NODE PROCESSING MODE

Name	Measure	Function
svavail	Service availability	if ((#$P_{\text{d-idle}}$==1 or #$P_{\text{d-off}}$==1) and (#$P_{\text{n-idle}}$==1 or #$P_{\text{n-run}}$==1) and (#$P_{\text{l-up}}$==1)) then 1 else 0
sysavail	System availability	if ((#$P_{\text{d-idle}}$==1 or #$P_{\text{d-off}}$==1) and (#$P_{\text{n-idle}}$==1 or #$P_{\text{n-run}}$==1)) then 1 else 0
svthru	Service throughput	prob(#$P_{\text{n-run}}$==1) * ν_n

D. Environmental model

In the above two models, the job arrival rate γ and the link failure rate λ_l are dependent on environmental uncertainties.

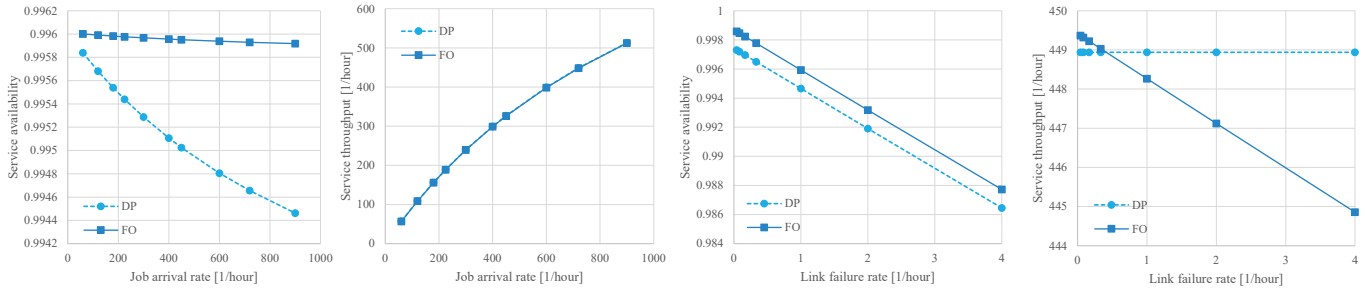


(a) Workload intensity model

(b) Communication quality degradation model

Figure 5. SRN for environmental uncertainties

In order to incorporate the environmental uncertainty factors, we define two environmental models that represent the states of the workload and communication link quality, as shown in Figure 5. The workload intensity model captures the variation of workloads depending on the mission status of the drone application (see Figure 5(a)). Each marking in this subnet corresponds to a state of workload that is associated with the job arrival rate γ. For example, if a token is deposited in P_{w0}, the corresponding job arrival rate γ_0 is assigned to $T_{\text{d-job}}$ in the drone models. We assume that the making #P_{w0}==1 represents the



(a) Service availability under varying workload (b) Service throughput under varying workload (c) Service availability under varying link reliability (d) Service throughput under varying link reliability

Figure 6. Sensitivity analysis results on service availability and service throughput by varying workload and link reliability

lowest workload state, while the making #P_{wn}==1 represents the highest workload state. The workload state transitions occur between the lowest and the highest cases. We assume that the average transition times for T_{wij} are given by $1/\beta_{ij}$.

The communication quality degradation model represents the changes of the communication link quality, which depends on the location of the drone. Like the workload intensity model, each marking in the subnet of Figure 5(b) corresponds to a state of communication link that is associated with the link failure rate λ_l. We assume that the making #P_{l0}==1 represents the most robust state, while the making #P_{wn}==1 represents the most unreliable state. The communication quality state transitions occur between these states. We assume that the average transition times for T_{lij} are given by α_{ij}.

VI. Numerical Experiments

In order to show the effectiveness of PA-offload, we conduct numerical experiments using the performability model constructed in Section V.

A. Experimental configuration

TABLE IV shows the parameter values used in numerical experiments. The values are chosen arbitrary from realistic ranges of values under reasonable constraints described below.

TABLE IV. Parameters for Numerical Experiments

Variable	Description	Value [1/hour]
γ	Job arrival rate	720
ν_d	Service rate on drone	1200
ν_n	Service rate on fog node	1440
λ_d	Process failure rate on drone in idle state	0.002976190
λ_{d2}	Process failure rate on drone in processing state	0.013888889
μ_d	Process recovery rate on drone	3
λ_n	Process failure rate on fog node in idle state	0.000462963
λ_{n2}	Process failure rate on fog node in processing state	0.001388889
μ_n	Process recovery rate on drone	2
ω	Communication rate between drone and fog node	7200
λ_l	Communication link failure rate	0.5
μ_l	Communication link recovery rate	360

While the mean time between failures (MTBF) of the drone is expected to be over a few ten thousands of hours [9], software execution errors considered in this paper may occur more frequently. Since the process failure rate is also affected by the state of the process, we set parameter values under the condition $\lambda_d < \lambda_{d2}$ and $\lambda_n < \lambda_{n2}$. We also assume that $\lambda_n < \lambda_d$ and $\lambda_{n2} < \lambda_{d2}$ as the fog node provide more robust execution environment than the drone. For the service rate, we assume $\nu_n > \nu_d$, since the fog node provides more resources, while the fog offloading incurs the additional communication delay $1/\omega$.

Both the drone processing model and the fog offloading model are implemented using SPNP [7]. We used the numerical solution with Gauss-Seidel method to compute the expected reward rates in steady-state.

B. Sensitivity analysis on workload variation

First, we evaluate the impacts of the workload on the service availability and performance of the drone processing mode (DP) and the fog offloading mode (FO). In this experiment, we do not use the environmental model shown in Section V-D. Instead, we fix the communication link failure rate λ_l to 1 and vary the job arrival rate γ in a range of [60, 900] (1/hour). The expected system availabilities and service throughputs are plotted in Figure 6 (a) and (b), respectively.

As the job arrival rate increases, service availability decreases monotonically, while the service throughput increases. Regardless of the job arrival rate, fog offloading is always preferable in terms of service availability since the fog node provides more reliable computing resource. Although the difference is not significant in a low workload condition, the degradation of the service availability due to an increased workload in DP may become a matter. For the service throughput, there is no significant difference between modes. In our job service model, we do not consider request buffer or parallel execution, and hence a new job is simply dropped if another job exists in the system. SRN models can be extended to incorporate buffering and parallel execution modes. However, in this work, we focus on a simple service model to compare the performance of DP and FO.

C. Sensitivity analysis on link reliability

Next, we evaluate the impacts of the communication link reliability on the service availability and the performance. In this experiment, we fix the job arrival rate γ to 720 (1/hour), and vary the communication link failure rate λ_l in a range of [0.04, 4] (1/hour). The expected system availabilities and service throughputs are plotted in Figure 6 (c) and (d), respectively.

As can be seen, the service availability decreases monotonically by increasing the link failure rate in both modes.

TABLE VI. EXPECTED SERVICE THROUGHPUT AND SYSTEM AVAILABILITY UNDER VARYING ENVIRONMENTAL STATES

Workload state		Low			Medium			High		
Link state		Robust	Degrade	Unstable	Robust	Degrade	Unstable	Robust	Degrade	Unstable
Service throughputs	DP	**108.946**	**108.946**	**108.946**	**276.414**	**276.414**	**276.414**	448.939	**448.939**	**448.939**
	FO	108.8811	108.6592	107.2026	276.3876	275.8408	272.252	**449.125**	448.2643	442.6149
System availability	DP	0.998448	0.998448	0.998448	0.997941	0.997941	0.997941	0.997419	0.997419	0.997419
	FO	**0.99876**	**0.99876**	**0.99876**	**0.99873**	**0.99873**	**0.99873**	**0.9987**	**0.9987**	**0.9987**

The results are reasonable, since the communication link availability is a composite part of the service availability. FO always achieves higher availability in this domain as well. For the service throughput, however, DP has an advantage as it is not affected by the change of communication link quality. In terms of service throughput, FO is effective only when the communication link is stably reliable.

D. Evaluation of PA-offload

As presented previously, both DP and FO have their own benefits under different environments and criteria (availability and performance). PA-offload can adapt to the change of environments and choose the best processing mode in terms of performability.

To evaluate the effectiveness of PA-offload, we incorporate the environmental model discussed in Section V-D. We assume a variation of the workload in three-stages: 0 (low), 1 (medium) and 2 (high), and assign to these states a corresponding job arrival rate of γ_0, γ_1 and γ_2, respectively. Similarly, the communication link quality is assumed to degrade in three-stages: 0 (robust), 1 (degrade), and 2 (unstable), which has associated link failure rate of λ_{l0}, λ_{l1}, and λ_{l2}, respectively. The parameter values and transition rates are presented in TABLE V.

TABLE V. PARAMETERS FOR ENVIRONMENTAL MODEL

Variable	Description	Value [1/hour]
γ_0	Job arrival rate in low workload	120
γ_1	Job arrival rate in medium workload	360
γ_2	Job arrival rate in high workload	720
λ_{l0}	Link failure rate in robust state	0.25
λ_{l1}	Link failure rate in degraded state	1
λ_{l2}	Link failure rate in unstable state	6
β_{01}, β_{12}	Transition rate to higher workload state	12
β_{10}, β_{21}	Transition rate to lower workload state	12
α_{01}, α_{12}	Transition rate to lower link reliability	2
α_{10}, α_{21}	Transition rate to higher link reliability	12

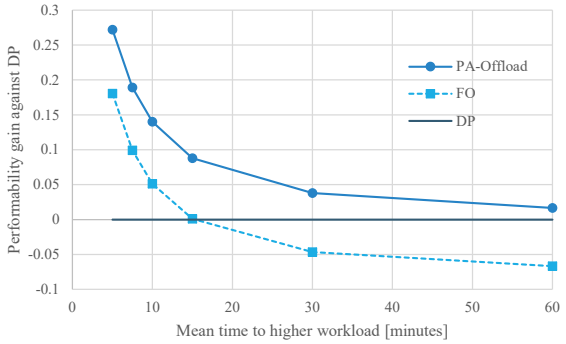
Each marking of the environmental model represents a specific environmental state that results in different system performance. In accordance with the definition of performability in Section IV-B, we compute the average service throughput as the concerned performability measure by

$$P_a = \sum_{m \in A} \tau(m)\pi(m),$$

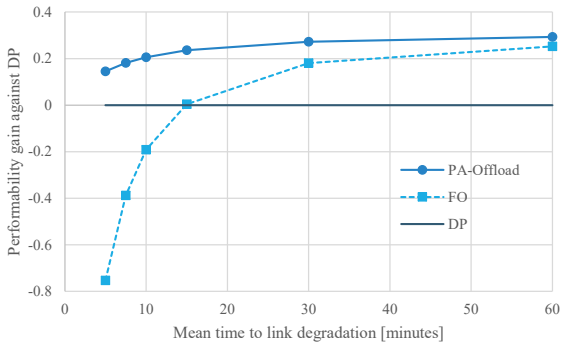
where $\tau(m)$ represents the service throughput for a making m, $\pi(m)$ is the steady-state probability for m, and A is the set of markings which satisfy the condition specified by the reward function *svavail*. Note that the performability is computed by a summation, since we deal with a discrete state-space here.

Given marking m, the job arrival rate and the link failure rate are determined. Table VI summarizes the expected service throughputs and system availabilities of DP and FO for the given environmental state. DP generally performs better in the service throughput except when the system is under high workload and the link condition is stable. In terms of system availability, however, FO is always better than DP. Under this circumstance, the decision of offloading is not straightforward.

PA-offload can determine the effective offloading condition by computing performability and comparing DP and FO. For the given environmental model, we vary the values of the workload amplification rates ($\beta_{01} = \beta_{12}$) and compute the performability gains against the case of DP mode. For example, the performability gain of FO against DP is computed by $E[P_{a,F}|\boldsymbol{\theta}] - E[P_{a,D}|\boldsymbol{\theta}]$. The results are shown in Figure 7 (a).



(a) Sensitivity to workload amplification rates



(b) Sensitivity to the link degradation rates

Figure 7. Performability gains by PA-offload and FO against DP by varying (a) workload intensities, and (b) link reliabilities

As can be seen, FO is preferable in terms of performability only when the mean time to higher workload is relatively short (<15 minutes). PA-Offload always achieves a higher performability when compared to DP and FO regardless of the values of workload amplification rates. We also compute the performability gains by varying the link reliability degradation rates ($\alpha_{01} = \alpha_{12}$) and show the results in Figure 7 (b). PA-Offload, in this case as well, always achieves the best performability results. FO can achieve higher performability

than DP only when the mean time to link degradation is larger than 15 minutes. With regarding the computation overhead, all availability computations are completed in less than a second using SPNP, thus we consider that the PA-Offload overhead is insignificant in terms of the decision mechanism.

From the experimental observation, we can summarize the advantage of PA-Offload in twofold. First, PA-offload achieves a desirable trade-off between service availability and performance. As Table VI shows, the decision of offload is not trivial under the competing availability and performance measures. The defined performability measure helps to decide the reasonable balance between them. Second, PA-offload always outperforms DP and FO in terms of the performability, since it can dynamically adapt the processing mode to changes in environmental states. Figure 7 shows that the benefit of PA-offload is consistent regardless of the environmental changes.

VII. Conclusion

This paper proposed PA-offload as an offloading decision scheme for drone image processing tasks in order to improve the performability of the system. The performability is defined as a composite measure of service throughput with system availability, so that it is used to make better performance-availability trade-offs by switching the computation mode. To compute the performability under uncertain environmental factors, we modeled the system behavior using SRNs. PA-offload can determine relevant states to start or stop the fog offloading by analyzing the expected performabilities under different modes for given environmental conditions. The numerical experimental results show that PA-Offload always achieves a better performability under varying workloads and link qualities during the drone operation.

Our study can be extended in the following directions. In our SRN model, we only consider a single drone system that consists of a drone, a client, a fog node, and a communication link. We can extend our model to larger systems that may consist of multiple drones and fog nodes by introducing additional SRN models. We can also consider the energy, the autonomy, and the safety (e.g., avoiding trashing) of drones as other important performance measures. Finally, the validation of the models is an important challenge in future works.

Acknowledgment

This work was supported in part by the grant of University of Tsukuba Basic Research Support Program Type S.

References

[1] B. Mishra, D. Garg, P. Narang, and V. Mishra, Drone-surveillance for search and rescue in natural disaster, Computer Communications, 2020.

[2] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar, UAVs for smart cities: Opportunities and challenges, In 2014 Int'l Conf. on Unmanned Aircraft Systems (ICUAS), pp. 267-273, 2014.

[3] N. Chen, Y. Chen, Y. You, H. Ling, P. Liang, and R. Zimmermann, Dynamic urban surveillance video stream processing using fog computing, In Int'l Conf. on multimedia big data, pp. 105-112, 2016.

[4] A. Yousefpour, et al., All one needs to know about fog computing and re lated edge computing paradigms: A complete survey, Journal of Systems Architecture, vol. 98, pp. 289-330, 2019.

[5] K. S. Trivedi, Probability and statistics with reliability, queuing, and com puter science applications, John Wiley, New York, 2001.

[6] J. F. Meyer, On evaluating the performability of degradable computing systems, IEEE Transactions on computers, no. 8, pp. 720-731, 1980.

[7] G. Ciardo, J. K. Muppala, K. S. Trivedi et al., Spnp: Stochastic petri net package. In Int'l Workshop on Petri Nets and Performance Models, vol. 89, 1989, pp. 142-151.

[8] R. A. Sahner, K. Trivedi, and A. Puliafito, Performance and reliability analysis of computer systems: an example-based approach using the SHARPE software package, Springer Science & Business Media, 2012.

[9] E. Petritoli, F. Leccese, and L. Ciani, Reliability and maintenance analysis of unmanned aerial vehicles, Sensors, vol. 18, no. 9, p. 3171, 2018.

[10] G. Ciardo, A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi, Automated generation and analysis of markov reward models using stochastic reward nets, In Linear Algebra, Markov Chains, and Queueing Models. Springer, pp. 145-191, 1993.

[11] Y. Zhang, H. Liu, L. Jiao, and X. Fu, To offload or not to offload: an efficient code partition algorithm for mobile cloud computing, In 1st Int'l Conf. on Cloud Networking (CLOUDNET), pp. 80-86, 2012.

[12] P. Mach and Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1628-1656, 2017.

[13] M. Deng, H. Tian, and B. Fan, Fine-granularity based application offloading policy in cloud-enhanced small cell networks, In IEEE Int'l Conf. on Communications Workshops (ICC), pp. 638-643, 2016.

[14] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, Delay-optimal computation task scheduling for mobile-edge computing systems, In IEEE Int'l Symp. on Information Theory (ISIT), pp. 1451-1455, 2016.

[15] Y. Mao, J. Zhang, and K. B. Letaief, Dynamic computation offloading for mobile-edge computing with energy harvesting devices, IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3590-3605, 2016.

[16] M. Kamoun, W. Labidi, and M. Sarkiss, Joint resource allocation and offloading strategies in cloud enabled cellular networks, In IEEE Int'l Conference on Communications (ICC). pp. 5529-5534, 2015.

[17] W. Labidi, M. Sarkiss, and M. Kamoun, Energy-optimal resource scheduling and computation offloading in small cell networks, In 22nd Int'l Conf. on telecommunications (ICT), pp. 313-318, 2015.

[18] M. H. Chen, B. Liang, and M. Dong, A semidefinite relaxation approach to mobile cloud offloading with computing access point, In Int'l Works. on Signal Processing Advances in Wireless Comm., pp. 186-190, 2015.

[19] O. Munoz, A. Pascual-Iserte, and J. Vidal, Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading, IEEE Trans. on Vehicular Technology, vol. 64, no. 10, pp. 4738-4755, 2014.

[20] X. Hou, Z. Ren, J. Wang, S. Zheng, W. Cheng, and H. Zhang, Distributed fog computing for latency and reliability guaranteed swarm of drones, IEEE Access, vol. 8, pp. 7117-7130, 2020.

[21] 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.

[22] D. Chemodanov, C. Qu, O. Opeoluwa, S. Wang, and P. Calyam, Policy-based function-centric computation offloading for real-time drone video analytics, In IEEE Int'l Symp. on Local and Metropolitan Area Networks (LANMAN), pp. 1-6, 2019.

[23] K. S. Trivedi, J. K. Muppala, S. P. Woolet, and B. R. Haverkort, Composite performance and dependability analysis, Performance Evaluation, vol. 14, no. 3-4, pp. 197-215, 1992.

[24] Y. Ma, J. J. Han, and K. S. Trivedi, Composite performance and availability analysis of wireless communication networks, IEEE Trans. on Vehicular Technology, vol. 50, no. 5, pp. 1216-1223, 2001.

[25] F. Machida, R. Xia, and K. S. Trivedi, Performability modeling for RAID storage systems by markov regenerative process, IEEE Trans. on Dependable and Secure Computing, vol. 15, no. 1, pp. 138-150, 2015.

[26] Z. Bakhshi, G. Rodriguez-Navas, and H. Hansson, Dependable fog computing: A systematic literature review, In 45th Euromicro Conf. on Software Engineering and Advanced Applications, pp. 395-403, 2019.

[27] E. Andrade, B. Nogueira, I.de Farias Jr., and D. Araujo, Performance and availability trade-offs in fog-cloud IoT environments, Journal of Network and System Management, vol. 29, no. 2, 2021.