Drone-Hosted Computation for Emergency Response

Otto B. Piramuthu and Matthew Caesar

***Abstract*—Ad hoc computing needs at remote locations or locations with insufficient resources due to reasons such as natural disasters require flexible solutions that are readily deployed on demand. We consider one such scenario where unmanned aerial vehicles (UAVs) or drones can be used to provide necessary coverage in terms of computational support. Specifically, we consider an environment where ground-based computational demand is satisfied by aerial drones that share the computational load to provide seamless service. We study transfer and location policies where transfer policy determines whether a job is locally processed by the drone that receives the order and location policy determines where a job is processed if it is sent to another drone. Our results indicate that the mean queue length of jobs waiting to be processed at the drones decreases with sharing the job processing load among the drones in the modeled system. Our results also highlight the beneficial aspects of the two step transfer and location policies.**

***Index Terms*—Load sharing, location policy, transfer policy, unmanned aerial vehicle (UAV).**

I. Introduction

AN OVERWHELMING majority of global computational needs arise in environments that are resourceful and well-connected. These needs are generally satisfied in real-time through appropriate ground-based computational resources. While such scenarios are the norm, there exist scenarios where computational needs are not necessarily met in a timely manner due to access-related issues or the sheer unavailability of necessary resources within reach. Oftentimes, such scenarios occur in remote or rural locations [12] that are disconnected and are father away from *civilization* due to any number of reasons that include location characteristics (e.g., remote field sites [9], rugged terrain [21]), natural disasters that temporarily cut off an area and render it devoid of necessary computational resources [23], among others. The need for computational resources under such conditions are not necessarily any less significant since real-time computational access may be needed to remotely control critical sensitive operations or to quickly process necessary tasks in emergency situations. Recent developments in edge computing allows for the possibility to address such needs in a timely manner.

Manuscript received 2 November 2022; revised 9 March 2023; accepted 5 June 2023. Date of publication 8 June 2023; date of current version 21 November 2023. *(Corresponding author: Otto B. Piramuthu.)*

The authors are with the Department of Computer Science, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (e-mail: obp2@illinois.edu; caesar@illinois.edu).

Digital Object Identifier 10.1109/JIOT.2023.3284045

Advances in unmanned aerial vehicle (UAV) or drone technology in terms of their speed and flexible mobility that avoids ground-based obstructions as well as their on-board computational resources that include memory, storage, and processing power help provide real-time computational resources when and where needed [16]. For example, data generated at such locations can be processed close to their source to help facilitate the generation of reliable and timely actionable intelligence. When computational service needs are minimal, the provision of such a service by a single drone with enough on-board computational resources and battery power to last the duration of the service is relatively straightforward. Fairly standard heuristics [13] can be used to manage the drone-level queue such as earliest deadline first (EDF) to minimize the number of late tasks and shortest job first (SJF) to minimize mean flow completion time. However, any realistic scenario comes with complicated computational requirements that necessitate the use of multiple drones that must act in consort to reliably provide the required service. We study such a scenario and consider a few different solution options.

We study demand generated by ground-based Internet of Things (IoT) devices that are isolated from the rest of the world in terms of connectivity and computational resources. Since computational demand in such a system easily exceeds what can be provided by a single drone, we consider a set of connected drones that fly to the service demand location and hover over the demand site to provide computational service as required. While job scheduling heuristics, such as EDF and SJF, work perfectly well in a single server queue scenario, performance breaks down [15] when such heuristics are used to support job queues at each of the drones where jobs that could be processed at another drone are immediately transferred. To accommodate such a load sharing setup, we consider a bi-level [6] decision problem that includes a transfer threshold policy and a location policy for each job that is processed by the drone-based system. The transfer policy determines where (locally at that drone or remotely at another drone) to process a job and the location (e.g., random, threshold, and shortest queue) policy determines which drone processes a job. We follow the basic setup as in [6] and then tailor the analysis to drone-based systems.

The remainder of this article is organized as follows. We provide a brief review of related literature in Section II. We discuss the considered load sharing policies in Section III. Next, we evaluate these policies and discuss results in Section IV. We then conclude this article in Section V.

II. Related Literature

Researchers have studied various facets of UAV use to communicate with ground-based IoT devices to deliver a solution that is generally competitive with purely ground-based solutions. For example, Bushnaq et al. [3] studied a UAV-based framework for wildfire detection in which they distribute a large number of low-cost self-powered IoT devices with sensors that detect fire across the forest of interest. When fire is detected, these sensors transmit their readings to UAVs that hover over the forest. They use discrete-time Markov chain to optimize the density of deployed IoT devices in the area of interest as well as the number of deployed UAVs. Through fire detection and false alarm probabilities, they find that their method offers a faster and reliable wildfire detection solution. In a similar vein, Hoque et al. [14] developed a UAV-based IoT as a service (IoTaaS) framework. They consider applications with heterogeneous sensor-based IoT devices that are placed in constrained or physically inaccessible area to monitor environmental conditions that are then reported to UAVs that hover around that general physical area. As proof-of-concept, they illustrate their developed framework for air pollution monitoring as a part of smart agriculture concept. Yang et al. [24] studied the use of UAVs as multihop relays in areas that are hard to access for rescue operations. Each of the UAVs in this network is connected to the gateway through a network of relay UAVs. They develop a centralized greedy algorithm and a distributed self-organized setup that help reduce the number of UAVs used and show that the self-organized setup is better in practice since it eliminates the need for global message exchange.

Nguyen et al. [19] studied the use of long-term evolution (LTE) signals for command and control of UAVs and observe that interference significantly affects coverage at high altitudes due to path loss becoming close to free space. To address such issues related to coverage and reliability, they recommend the simultaneous use of two independent networks.

Guo et al. [8] studied mobile UAV-based systems with the consideration of real-time self-monitored UAV battery power, and dynamic changes in service requirement, location, and demand. They use backpropagation-trained neural network to predict service requirement, location, and demand to provide necessary service with the appropriate number of UAVs. The UAVs with drained battery-based power are switched with other UAVs with enough battery-based power to ensure seamless uninterrupted service while balancing (UAV-based) supply and (ground-based) demand. Similarly, Hammami et al. [10] studied intelligent monitoring through reinforcement learning of small cells mounted on UAVs that serve subscribers. These UAVs associate and dissociate with a set of Macrocell infrastructure depending on several factors that include their battery power levels and data demand. The proposed cooperative multiagent reinforcement approach is applied as an offline exploration step in which the UAVs learn to react and move as per bandwidth demands and an online step in which the UAVs instantaneously adapt to peak load scenarios. Sun et al. [22] also studied UAVs that act as drone small cells (DSCs) to complement traditional terrestrial small cells to provide connectivity across applications and incorporate both line-of-sight and non-line-of-sight components in both drone- and terrestrial-based scenarios. They evaluate network performance in terms of coverage probability, area spectral efficiency (ASE) and user throughput to quantify the impact of bias factor, DSC height, base station density on the network performance in terms of coverage probability, ASE, and average user throughput. Their results show that network performance deteriorates when both drone- and terrestrial-based cases are combined in ultra-dense networks.

Aissa et al. [1] modeled a cluster of UAVs with a cluster head UAV as the interface between demand from outside and the UAVs in that cluster. They use minimum cost flow algorithm for load balancing, offload path selection, and offload coordination. When the cluster head receives a service request, it probes the UAVs in the cluster for their remaining capacities. A route discovery protocol and load balancing algorithm are then used to determine the work sent to each UAV to minimize delay and to extend the UAV-based network's lifetime.

Apostolopoulos et al. [2] studied a scenario where the users can choose to offload data to a multi-access edge computing (MEC) environment comprising ground- and UAV-based MEC servers. The authors use prospect theory noncooperative game to model risk-aware user behavior in their decision-making process. Specifically, to maximize their satisfaction, the users can choose to offload a certain fraction of their data to UAV-mounted MEC and the rest to the ground-based counterpart where the former provides superior but uncertain results and the latter provides safe and guaranteed results. They model heterogeneous users in terms of their risk averseness and observe that the loss-averse users process more data locally in the ground-based system and experience less perceived satisfaction utility. They also observe that user heterogeneity increases UAV-mounted MEC servers' probability of failure.

Fan and Ansari [7] modeled UAVs that flexibly move around and relay signals between IoT devices that are present at high-densities and appropriate base stations to decrease communication latency in such IoT networks. Signal latency in this setup is determined by the access link between IoT devices and UAVs and the backhaul link between UAVs and base stations. They use heuristics to locate UAVs based on traffic demand distribution and then determine the best base station to associate with through a user association algorithm. Zhang and Ansari [26] considered a similar setup with the goal of minimizing the operating cost.

Hayajneh et al. [11] studied drone-empowered small cellular networks (DSCNs) with a partially operational ground cellular network in a post-disaster [25] scenario. Specifically, they consider design parameters, such as optimal altitude and number of UAV base stations as a function of destroyed base stations and associated signal propagation conditions. With coverage probability of a down-link mobile user as the performance metric, they show that by intelligently selecting the number of UAVs and their corresponding altitudes, service to ground-based mobile users is significantly enhanced without incurring significant performance penalty. Similar to Deruyck et al. [5] and Lyu et al. [17], Patra et al. [20] considered the use of

UAV-mounted mobile base stations to provide wireless coverage when cellular infrastructure is unavailable due to natural disasters. Each UAV serves a hotspot cell, which is a circular area comprising users with wireless service needs where the goal is to serve the highest number of user demand with minimal service interruption. The mobile users are allowed to switch between adjacent UAVs to balance supply and demand. They use packet delivery ratio, data served per unit of depleted energy, service discontinuity time, and the total amount of data served as performance criteria.

Based on our review of extant literature in this general area, we observe that none of the publications consider load sharing among a set of UAVs when the number of jobs to be processed at any time point exceeds their queue buffer limit for some or all of the UAVs. When the number of jobs to be processed at any time exceed the queue size for *every* UAV in the system, this signifies the need for more UAVs to be deployed since the demand far exceeds supply. However, when the number of jobs at only some UAVs exceed their allowed queue length, these *overflow* jobs need to be reallocated to other UAVs with fewer jobs waiting in queue to be processed. We consider this scenario where load sharing occurs among the UAVs.

III. Considered Policies

A natural disaster has the potential to damage critical communication infrastructure [16]. This is especially critical in remote areas with the need for computational resources. Under such circumstances, it is necessary to respond with appropriate resources as quickly as possible to minimize the resulting disruption. While installation of WiFi access points on site is a possibility, it requires significant manual labor and takes additional time during which all essential computations are inevitably put on hold. UAVs offer a better option to provide on-demand computation and communication close to where needed until a long-term solution is put in place. Since UAVs are resource constrained, there is a need for multiple load-sharing UAVs that work in consort to provide necessary services in a timely manner through operationalization of appropriate policies.

We study *random*, *threshold*, and *shortest* policies. In the *random* policy, a drone with a queue that exceeds a preset threshold (T) randomly selects another drone and transfers the next arriving job. We assume the same queue length threshold (T) for all the drones in the system. This assumption is valid for the scenario in which the system includes drones of the same type and model. In the *threshold* policy, a drone that wants to transfer a job to another drone upon its queue length reaching threshold T polls other drones and picks the first one that has not reached its queue length threshold limit. In the *shortest* policy, the drone that wants to transfer a job to another drone upon its queue length reaching threshold T polls a random set of other drones for their queue lengths and picks the one with the shortest queue length to transfer the job.

We use the following notation.

n	queue length at a drone.
T	queue length threshold at each drone.
L_t	static transfer limit.
L_p	static probe limit.
λ	average job arrival rate.
λ_t	unconditional job arrival rate.
$\lambda_t(n)$	conditional arrival rate of transferred jobs.
S	average service time per job.
ρ	load factor at each drone ($= \lambda S$).
C	cost to transfer a job to another drone.
K	number of drones in the system.
p_n	conditional probability of n ($0 \le n \le T$).
p_T^a	absolute probability of being in a transferring phase.
$\tilde{n}_p$	mean queue length with node in processing phase.
$pshort_n$	probability that queue length n is chosen.

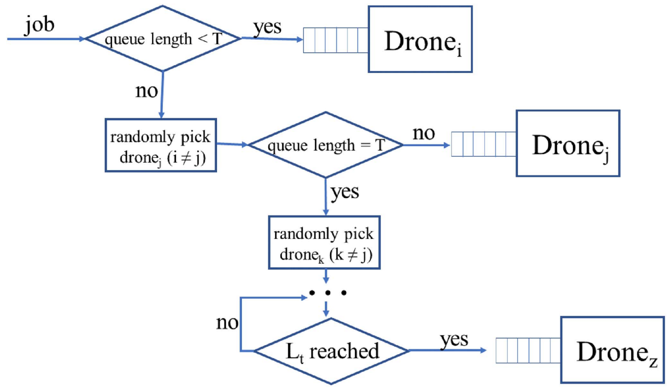

Fig. 1. Random policy.

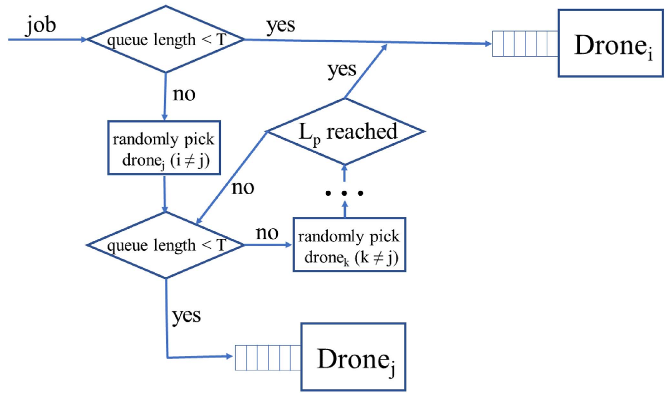

Fig. 2. Threshold policy.

In the *random* policy (Fig. 1), when the static transfer limit (L_t) is reached, that job can no longer be transferred and must be processed at the chosen drone. Similarly, when the static probe limit (L_p) is reached for the *threshold* policy (Fig. 2), the source drone for that job must process the job.

The constants T, L_p, and L_t can be chosen based on the specific system characteristics. For example, T is upper bounded by the drone's buffer size. Similarly, the upper limit for L_p is $K - 1$ and that for L_t is $K - 1$ if a drone is at the receiving end of a given job only once.

In the *shortest* policy (Fig. 3), L_p drones are randomly probed for their queue lengths and the shortest among those is chosen as long as this shortest queue length is less than threshold T.

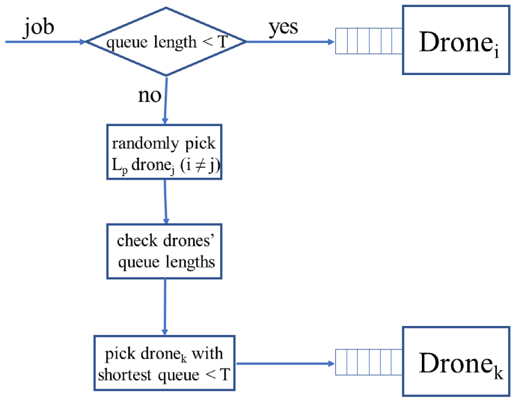
job
queue length < T
yes
Drone$_i$
no
randomly pick L$_p$ drone$_j$ ($i \neq j$)
check drones' queue lengths
pick drone$_k$ with shortest queue < T
Drone$_k$

Fig. 3. Shortest policy.

The drone that transfers a job incurs related processing cost, which is allocated to the source drone. The processing cost at the receiving drone is included in the service time (S) for that job. We assume the cost for communication among drones to be negligible. This assumption is justified since the processing and waiting times at the drones far exceed the communication time among the drones [4], [18]. We model the system as a Markov model where the state of each drone is independent of the states of all other drones in the system. At any time, a drone is in one of two states: 1) *processing* when the job queue length is at most the threshold T or 2) *transferring* when a new job arrival temporarily results in the job queue length exceeding the threshold T. Each drone receives jobs to be processed from two sources that include new jobs and jobs that are transferred from other drones. The total arrival rate at each drone is therefore $\lambda + \lambda_t(T^+)$ where $\lambda_t(T^+)$ represents that which is transferred from another drone with queue length more than the threshold T. We now model each of the considered policies.

A. Random Policy

When following the *random* policy, the source drone transfers jobs with no regard for the state (i.e., queue length) of the destination drone. Therefore, the queue length (n) at the destination drone is irrelevant

$$\lambda_t(n) = \lambda_t. \tag{1}$$

Since under the *random* policy, all jobs that arrive after the queue length reaches threshold T are transferred, except those that reach transfer limit L_t, the rate of job transfer equals the arrival rate of transferred jobs. The expression therefore includes terms for the rate at which each of the jobs are transferred

$$\lambda_t = \lambda\left(\sum_{l=1}^{L_t}\left(p_T\left(1-p_T^a\right)+p_T^a\right)^l\right).$$

Upon summation of the above expression, we get

$$\lambda_t = \left(p_T\left(1-p_T^a\right)+p_T^a\right)\frac{\lambda\left(1-\left(p_T\left(1-p_T^a\right)+p_T^a\right)^{L_t}\right)}{\left(1-\left(p_T\left(1-p_T^a\right)+p_T^a\right)\right)}. \tag{2}$$

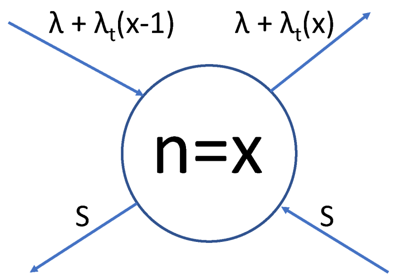

Fig. 4. Birth–death model.

The absolute probability of being in a transferring phase is related to the drone processing power utilization to accomplish job transfers and processing those jobs that cannot be transferred to other drones due to reaching the transfer limit

$$p_T^a = C\lambda_t + S\lambda\left(p_T\left(1-p_T^a\right)+p_T^a\right)^{L_t+1}.$$

Incorporation of (2) in the above expression results in

$$p_T^a = \frac{p_T S(\lambda+\lambda_t) - (S-C)\lambda_t}{1-(1-p_T)S(\lambda+\lambda_t)}. \tag{3}$$

Based on the solution to the birth–death model (Fig. 4) and expression (1), we get

$$P_T = \frac{(S(\lambda+\lambda_t))^T(1-S(\lambda+\lambda_t))}{1-(S(\lambda+\lambda_t))^{T+1}}. \tag{4}$$

From expressions (2), (3), and (4), we can determine p_T^a. Next, we can determine the conditional mean queue length based on the birth–death model solution

$$\tilde{n}_p = S(\lambda+\lambda_t)\left(\frac{1-\rho-C\lambda_t}{1-p_T^a}\right) \times \left(\frac{1-(S(\lambda+\lambda_t))^T}{(1-S(\lambda+\lambda_t))^2} - \frac{T(S(\lambda+\lambda_t))^T}{1-S(\lambda+\lambda_t)}\right). \tag{5}$$

B. Threshold Policy

The *threshold* policy allows for transfer of jobs to a drone only when its queue length is less than threshold T, which signifies that both $\lambda_t(T)$ and $\lambda_t(T^+)$ are zero. However, when the drone's queue length has not reached threshold T, the transferred job arrival rate is independent of the drone's state. For $0 \le n \le (T-1)$

$$\lambda_t(n) = \lambda_t^*. \tag{6}$$

The probability that the queue length at a drone is less than the threshold T is given by $(1-p_T)(1-p_T^a)$, we have

$$\lambda_t^* = \frac{\lambda_t}{(1-p_T)\left(1-p_T^a\right)}. \tag{7}$$

Since under the *threshold* policy, all jobs that arrive after the queue length reaches threshold T are transferred, except those that reach the probe limit L_p, the rate of job transfer equals the arrival rate of transferred jobs. The expression therefore includes terms for the rate at which each of the jobs are transferred

$$\lambda_t = \left(p_T\left(1-p_T^a\right)+p_T^a\right)\lambda\left(1-\left(p_T\left(1-p_T^a\right)+p_T^a\right)^{L_p}\right).$$

Based on (7) and the above expression, we get

$$\lambda_t^* = \frac{\left(p_T\left(1-p_T^a\right)+p_T^a\right)\lambda\left(1-\left(p_T\left(1-p_T^a\right)+p_T^a\right)^{L_p}\right)}{\left(1-p_T\right)\left(1-p_T^a\right)}. \quad (8)$$

The absolute probability of being in a transferring phase is related to the drone processing power utilization to accomplish job transfers and processing those jobs that cannot be transferred to other drones due to failure to identify an appropriate drone to process such jobs

$$p_T^a = C\lambda_t + S\lambda\left(p_T\left(1-p_T^a\right)+p_T^a\right)^{L_p+1}.$$

Based on (8) and the expression above, we get

$$p_T^a = \frac{p_T S\lambda - (1-p_T)(S-C)\lambda_t^*}{1-(1-p_T)\left(S\lambda + (S-C)\lambda_t^*\right)}. \quad (9)$$

Based on the solution to the birth–death model (Fig. 4) and expression (6), we have

$$p_T = \frac{\left(S\left(\lambda+\lambda_t^*\right)\right)^T\left(1-S\left(\lambda+\lambda_t^*\right)\right)}{1-\left(S\left(\lambda+\lambda_t^*\right)\right)^{T+1}}. \quad (10)$$

From expressions (7), (8), (9), and (10), we can determine p_T^a. We can then determine the conditional mean queue length based on the birth–death model solution

$$\tilde{n}_p = S\left(\lambda+\lambda_t^*\right)\left(\frac{1-\rho-C\lambda_t}{1-p_T^a}\right) \times \left(\frac{1-\left(S\left(\lambda+\lambda_t^*\right)\right)^T}{\left(1-S\left(\lambda+\lambda_t^*\right)\right)^2} - \frac{T\left(S\left(\lambda+\lambda_t^*\right)\right)^T}{\left(1-S\left(\lambda+\lambda_t^*\right)\right)}\right). \quad (11)$$

C. Shortest Policy

In the *shortest* policy, as with the *random* and *threshold* policies, when the number of jobs waiting in queue to be processed at a given drone reaches the threshold T, that drone attempts to transfer all new job arrivals to other drones for processing until its queue length drops below T, which signals that additional jobs can be placed in queue for processing at that drone. The drone probes the queue length of a selected number (L_p) of other drones and chooses the drone with the shortest queue to transfer that job for processing, with the condition that the queue length at that destination drone is below the threshold T.

Similar to the *threshold* policy, all jobs that arrive after the queue length reaches threshold T are transferred, except those that reach the probe limit L_p, the rate of job transfer equals the arrival rate of transferred jobs. The expression therefore includes terms for the rate at which each of the jobs are transferred

$$\lambda_t = \left(p_T\left(1-p_T^a\right)+p_T^a\right)\lambda\left(1-\left(p_T\left(1-p_T^a\right)+p_T^a\right)^{L_p}\right).$$

Again, similar to that in the *threshold* policy scenario, the absolute probability of being in a transferring phase is related to the drone processing power utilization to accomplish job transfers and processing those jobs that cannot be transferred to other drones due to failure to identify an appropriate drone to process such jobs

$$p_T^a = C\lambda_t + S\lambda\left(p_T\left(1-p_T^a\right)+p_T^a\right)^{L_p+1}.$$

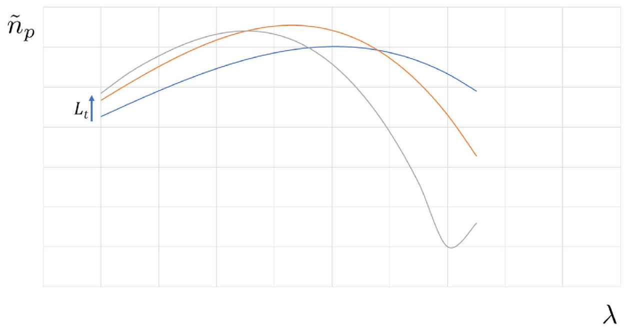

Fig. 5. [Random policy] mean queue length versus λ.

In the *shortest* policy case, let $pshort_n$ $(0 \le n \le T-1)$ be the probability that drone with queue length n is chosen for job transfer. $pshort_n$ can be written in terms of the probe limit (L_p), p_n the conditional state probabilities, and P_T^a. Specifically, the expression includes each of the L_p randomly chosen drones with a queue length of at least n. This can be determined through subtracting the probability of those with queue length at least $n+1$ from those with queue length of at least n since we want to choose the shortest queue length

$$pshort_n = \left(1-\left(\sum_{m=0}^{n-1} p_m\right)\left(1-p_T^a\right)\right)^{L_p} - \left(1-\left(\sum_{m=0}^{n} p_m\right)\left(1-p_T^a\right)\right)^{L_p}. \quad (12)$$

The arrival rates $(0 \le n \le T-1)$ can now be determined as

$$\lambda_t(n) = \frac{\lambda_t}{p_n\left(1-p_T^a\right)}\frac{pshort_n}{1-\left(p_T\left(1-p_T^a\right)+p_T^a\right)^{L_p+1}}. \quad (13)$$

From the birth–death model, we have

$$p_n = p_{n-1}\left(1+\frac{\lambda_t(n-1)}{\lambda}\right)\rho. \quad (14)$$

And

$$p_0 = \frac{1-\rho-C\lambda_t}{1-p_T^a}. \quad (15)$$

IV. Results

We first consider the dynamic between the mean queue length with node in processing phase ($\tilde{n}_p$) and the average job arrival rate (λ) for the *random* policy. We fix C, S, and T at constant values, vary the arrival rate, and observe the mean queue length of jobs waiting to be processed at the drones. We varied λ from 0.2 to 0.85 and observed $\tilde{n}_p$. We also considered three different cases where we set the transfer threshold at three different values. This essentially affects when jobs begin getting transferred from a drone. The result of this analysis is given in Fig. 5. As the job arrival rate is increased, we expected to witness a concomitant increase in the mean queue length. However, as can be seen in Fig. 5, the mean queue length increases to a certain level and then starts to decrease. This is the case for all three considered threshold

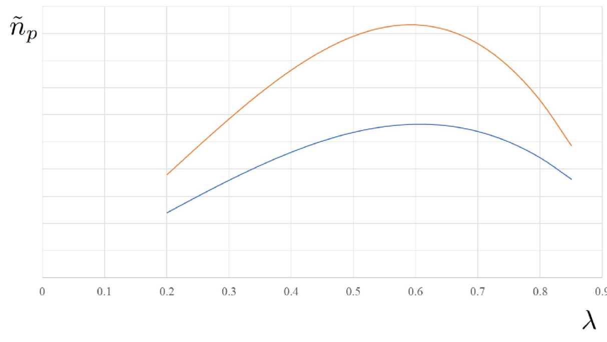

Fig. 6. [Threshold policy] mean queue length versus λ.

values. The gray, orange, and blue curves, respectively, represent highest to lowest transfer threshold values. We observe that the blue curve remains lower than the red curve until λ is about 0.68 when the red curve is lower than the blue curve. The maximum for the blue curve occurs at a higher λ value as compared to that of the red curve since job transfers begin to occur earlier in the lower threshold case. We also observe that the gray curve goes up toward the end when all the queues have enough number of jobs to be processed.

This signifies that the queue length increases as the arrival rate increases up to a certain extent, when the threshold is reached, and jobs begin to be transferred to other drones for processing. As the jobs are shared among the drones, on average, the queue length at the drones decreases. This likely happens as the effective number of jobs at each drone varies and some drones are under-utilized. Once job sharing occurs, the drones with more than average number of jobs waiting in queue transfer jobs to those drones with less than average number of jobs waiting in queue to be processed. Note that since we fix the number of drones in the system at K, the reason the average n does not increase with λ toward the end in all the shown (Figs. 5 and 6) cases is that we stopped the considered λ value at 0.85, which is a reasonable value for the modeled system.

We next considered a similar scenario for the *threshold* policy. The curves (Fig. 6) are very similar to that for the *random* policy, with the blue curve being lower than the red curve. The structure of the curves are also similar to those in the *random* policy case, which had both the curves in concave form. Since the curves for both *threshold* and *random* policies are comparable, the interpretation of these are similar. The queue length increases at the beginning as the arrival rate increases and turns around and starts to decrease as the queue length goes over the threshold in some of the drones that have high utilization and jobs are shared with those drones that have much shorter queue lengths.

Clearly, in the *shortest* policy case, the patterns observed with respect to average queue length and arrival rate in the *random* and *threshold* policy cases are only magnified. This is because the transferred jobs are specifically targeted to be sent to drones with the shortest queues, thereby distributing the workload among all the drones. While more work is done by the source drone to determine which drone to send the overflow jobs, the receiving drone's short queue length ensures that the transferred job gets processed sooner rather than later.

V. Discussion and Conclusion

The need for computational resources can arise anywhere, and such needs are relatively easy to satisfy for the most part except when the needed locations are farther away from the rest of the world. Such remote locations, possibly under rough terrain conditions, are often difficult to reach. However, accessibility issues do not signify the disappearance of such needs. Recent developments in UAV technology and the significant growth in UAV's popularity in a wide variety of application domains necessitates serious consideration of UAVs to provide necessary computational resources and support, especially at remote locations.

We considered the use of a set of drones at a remote location under disaster circumstances. In order to deliver appropriate help in a timely manner, there is a need to have sufficient and timely access to computational resources on hand. While a set of UAVs can deliver required service, the setup can quickly become complex when the workload demanded is high, thereby resulting in inefficient use of constrained resources. As UAVs operate with on-board batteries as the sole power source, there is a limit to available power at any UAV. Moreover, when processing jobs at such scenarios, there is bound to be imbalance in job arrivals at each of these UAVs.

To efficiently and effectively process all incoming jobs, we considered a two-stage decision-making setup where the first decision involves whether to process a job locally at a drone that first received that job or transfer that job to another drone for processing. The next stage, if a job is scheduled for transfer to another drone, is to decide which other drone should receive and process this overflow or transfer job. We considered three transfer policies and studied some of their characteristics. While this is a start, other transfer policies that are specifically designed for UAV load sharing would benefit from extensive analysis so real-world deployments can take place with a reasonably good understanding of these systems.

In addition to UAV-specific transfer policies, a possible extension is the use of cloud-based computational resources when some latency is not of concern. With this option, the tradeoffs between processing jobs on UAVs versus communication cost involved in terms of latency when cloud-based systems are invoked and the processing time at the cloud need to be considered. Clearly, the cloud-based option may not be attractive when the jobs can be quickly processed at the drones when the queues across the drones are rather short. A concern is the security of sensitive information processed in drones versus cloud-based systems. Another extension is dynamic allocation of drones to a site depending on expected load. The extensions would therefore depend on several factors, such as the urgency of the jobs to be processed, the expected load and associated frequency of job arrivals at the UAVs, individual security requirements for these jobs, and the security guarantees that can be provided by UAVs and cloud-based services.

References

[1] S. B. Aissa, A. B. Letaifa, A. Sahli, and A. Rachedi, "Computing offloading and load balancing within UAV clusters," in *Proc. IEEE 19th Annu. Consum. Commun. Netw. Conf. (CCNC)*, 2022, pp. 497–498.

[2] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 175–190, Jan. 2023, doi: 10.1109/TMC.2021.3069911.

[3] O. M. Bushnaq, A. Chaaban, and T. Y. Al-Naffouri, "The role of UAV-IoT networks in future wildfire detection," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 16984–16999, Dec. 2021.

[4] N. Cheng et al., "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 26–32, Aug. 2018.

[5] M. Deruyck, J. Wyckmans, L. Martens, and W. Joseph, "Emergency ad-hoc networks by using drone mounted base stations for a disaster scenario," in *Proc. IEEE 12th Int. Conf. Wireless Mobile Comput. Netw. Commun. (WiMob)*, 2016, pp. 1–7.

[6] D. L. Eager, E. D. Lazowska, and J. Zahorjan, "Adaptive load sharing in homogeneous distributed systems," *IEEE Trans. Softw. Eng.*, vol. SE-12, no. 5, pp. 662–675, May 1986.

[7] Q. Fan and N. Ansari, "Towards traffic load balancing in drone-assisted communications for IoT," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3633–3640, Apr. 2019.

[8] H. Guo, X. Zhou, Y. Wang, and J. Liu, "Achieve load balancing in multi-UAV edge computing IoT networks: A dynamic entry and exit mechanism," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 18725–18736, Oct. 2022.

[9] Y. Guo, I. Mohamed, O. Abou-Sayed, and A. Abou-Sayed, "Cloud computing and Web application-based remote real-time monitoring and data analysis: Slurry injection case study, Onshore USA," *J. Petrol. Expl. Prod. Technol.*, vol. 9, pp. 1225–1235, Jun. 2019.

[10] S. E. Hammami, H. Afifi, H. Moungla, and A. Kamel, "Drone-assisted cellular networks: A multi-agent reinforcement learning approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.

[11] A. M. Hayajneh, S. A. R. Zaidi, D. C. McLernon, and M. Ghogho, "Drone empowered small cellular disaster recovery networks for resilient smart cities," in *Proc. IEEE Int. Conf. Sens., Commun. Netw. (SECON Workshops)*, 2016, pp. 1–6.

[12] S. Helmer et al., "Bringing the cloud to rural and remote areas via cloudlets," in *Proc. 7th Annu. Symp. Comput. Develop. (ACM DEV)*, 2016, pp. 1–10.

[13] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proc. ACM SIGCOMM*, 2012, pp. 127–138.

[14] M. A. Hoque, M. Hossain, S. Noor, S. M. R. Islam, and R. Hasan, "IoTaaS: Drone-based Internet of Things as a service framework for smart cities," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 12425–12439, Jul. 2022.

[15] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *Proc. 6th ACM Symp. Cloud Comput. (SoCC)*, 2015, pp. 111–124.

[16] I. Lee, V. Babu, M. Caesar, and D. Nicol, "Deep reinforcement learning for UAV-assisted emergency response," in *Proc. 17th EAI Int. Conf. Mobile Ubiquitous Syst. Comput., Netw. Services (MobiQuitous)*, 2020, pp. 327–336.

[17] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim, "Placement optimization of UAV-mounted mobile base stations," *IEEE Commun. Lett.*, vol. 21, no. 3, pp. 604–607, Mar. 2017.

[18] Y. Miao, J. Xu, M. Chen, and K. Hwang, "Drone enabled smart air-agent for 6G network," in *Proc. IEEE Int. Conf. Commun.*, 2022, pp. 1–6.

[19] H. C. Nguyen, R. Amorim, J. Wigard, I. Z. Kovacs, and P. Mogensen, "Using LTE networks for UAV command and control link: A rural-area coverage analysis," in *Proc. IEEE 86th Veh. Technol. Conf. (VTC-Fall)*, 2017, pp. 1–6.

[20] A. N. Patra, P. A. Regis, and S. Sengupta, "Dynamic self-reconfiguration of unmanned aerial vehicles to serve overloaded hotspot cells," *Comput. Electr. Eng.*, vol. 75, pp. 77–89, May 2019.

[21] L. Ríos. "The last Internet dealers of rural Mexico." 2022. [Online]. Available: https://restofworld.org/2022/the-last-internet-dealers-of-rural-mexico

[22] H. Sun, X. Wang, Y. Zhang, and T. Q. S. Quek, "Performance analysis and cell association design for drone-assisted heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13741–13755, Nov. 2020.

[23] K. C. Ujjwal, S. Garg, J. Hilton, J. Aryal, and N. Forbes-Smith, "Cloud Computing in natural hazard modeling systems: Current research trends and future directions," *Int. J. Disast. Risk Reduct.*, vol. 38, Aug. 2019, Art. no. 101188.

[24] T. Yang, C. H. Foh, F. Heliot, C. Y. Leow, and P. Chatzimisios, "Self-organization drone-based unmanned aerial vehicles (UAV) networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.

[25] E. Yanmaz, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter, "Drone networks: Communications, coordination, and sensing," *Ad Hoc Netw.*, vol. 68, pp. 1–15, Jan. 2018.

[26] L. Zhang and N. Ansari, "Optimizing the operation cost for UAV-aided mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 6085–6093, Jun. 2021.