

Combined time bound optimization of control, communication, and data processing for FSO-based 6G UAV aerial networks

Seungwoo Seo  | Da-Eun Ko  | Jong-Moon Chung 

School of Electrical and Electronic Engineering, Yonsei University, Seoul, Rep. of Korea

Correspondence

Jong-Moon Chung, School of Electrical and Electronic Engineering, Yonsei University, Seoul, Rep. of Korea.
Email: jmc@yonsei.ac.kr

Funding information

This research was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00685, Free space optical communication based vertical mobile network).

Because of the rapid increase of mobile traffic, flexible broadband supportive unmanned aerial vehicle (UAV)-based 6G mobile networks using free space optical (FSO) links have been recently proposed. Considering the advancements made in UAVs, big data processing, and artificial intelligence precision control technologies, the formation of an additional wireless network based on UAV aerial platforms to assist the existing fixed base stations of the mobile radio access network is considered a highly viable option in the near future. In this paper, a combined time bound optimization scheme is proposed that can adaptively satisfy the control and communication time constraints as well as the processing time constraints in FSO-based 6G UAV aerial networks. The proposed scheme controls the relation between the number of data flows, input data rate, number of worker nodes considering the time bounds, and the errors that occur during communication and data processing. The simulation results show that the proposed scheme is very effective in satisfying the time constraints for UAV control and radio access network services, even when errors in communication and data processing may occur.

KEYWORDS

Apache Spark Streaming, free space optics, hierarchical unmanned aerial vehicle network, time bound optimization, unmanned aerial vehicle aerial network

1 | INTRODUCTION

As the amount of mobile traffic continuously increases, new expectations on 6G mobile network service types and performance bounds are being proposed. Among several candidate architectures of 6G mobile networks, the solution model using unmanned aerial vehicle (UAV)-based aerial platforms (APs) supporting multi-hop free space optical (FSO) broadband links [1] is a strong candidate. As a result, much research on AP-based FSO networking has been conducted. The AP network can provide an intermediate linkage between the base stations (BSs) to the core network

or the edge of the network. For wide outdoor service areas, it could be more cost-efficient to add on a UAV-based AP network than to build an additional wired overlaying the 6G mobile network. AP networks are also flexible in terms of the expansion and reduction of its scale so that it is possible to control the scalability according to mobile traffic demands. For this reason, 6G wireless networks based on multiple altitude APs formed by UAV groups have been considered in recent papers [2]. Unlike ground-based communication systems, APs require three-dimensional (3D) coordinates, which include global positioning system-based two-dimensional longitude and latitude information as well

as altitude information to express the position of the UAVs and their aerial motion vectors.

To satisfy the bandwidth requirements of a 6G wireless network [3], several studies have been conducted on multiple wireless signals. Considering the heavy future demands on 6G mobile network services, potential candidates that satisfy such signal conditions are FSO, radio frequency, mmWaves, lasers, and microwaves. Among these signals, FSO-based wireless communication has received much attention [4] because using FSO signals within infrared wavelength ranges enable high-quality point-to-point links from the ground to the UAV, and provides ultra-long connectivity between UAVs within the AP, especially for high-altitude platform (HAP)-based UAVs in the 17 km–22 km altitude range [2,5–7].

In particular, in APs, the channel environment continuously changes. Hence, in order for the UAVs to maintain reliable networking performance, it is crucial to analyze such variables in real-time and position the UAVs at the required locations in each AP, respective to the other UAVs and ground control/communication systems (GCSs) [8].

During communication in the AP network, either among UAVs or between a UAV and a BS, there are several factors that can degrade the system performance [9]. The main causes of system failure include misalignment, interference between the signal transmitter and receiver, atmospheric turbulence [10], reliability issues of the UAVs, and signal scattering at the stratospheric level.

In addition to UAV control and communication, data processing time is also important in the 6G UAV network. Because the UAV network will also have to support computing processes for real-time UAV control and network service support (including edge computing), real-time streaming-based big data technologies [11] such as Spark Streaming using mini-batches in the range of 0.5 s–3 s are needed. For this reason, spark streaming-based processing is applied as the real-time data processing scheme in this study. UAV control data as well as network service data are consistently transferred and processed through the UAV-based APs to maintain extension to the mobile radio access network, where predetermined time bounds have to be satisfied for real-time data [12].

The most significant challenge in this scheme is overcoming the erroneous events in real-time data processing that supports the UAV control and data services that have strict delay bounds. Because most big data platforms distribute tasks and decide the time bounds without considering the error rate, any error event in a single worker node might affect the entire communication system [13,14].

To overcome such problems, in this paper, a combined time bound optimization scheme is proposed that satisfies both control and communication time as well as the data processing time in FSO-based 6G UAV aerial networks.

The remainder of this paper is organized as follows. Section 2 briefly reviews studies related to UAV network optimization and Apache Spark Streaming. Section 3 proposes the system model used in this study. Section 4 introduces the proposed UAV network control system (NCS) and Spark Real-time Streaming Adaptive Failure-compensation (SRSAF) scheme used for combined time optimization. Section 5 explains the process of optimization using a flow-chart that performs the SRSAF combined time optimization process. Section 6 provides a performance analysis of the proposed SRSAF optimization process, and the paper is concluded in Section 7.

2 | BACKGROUND AND RELATED WORK

2.1 | UAV network optimization

As interest in UAV networks has increased, many papers describing techniques to support UAV networks have emerged. Most studies assume that the UAV plays the role of an intermediate node that connects the user equipment (UE) to the BS. To ensure that the UAVs perform their required functions smoothly, several studies have attempted to increase the stability of the network in terms of energy consumption, packet delay, and queuing [15–21].

Zhang and others [15] analyzed the response delay that occurs when a packet is transmitted in a two-layer UAV network by dividing it into two parts: communication and computation (queuing). The proposed two-layer UAV network consists of Bottom-UAVs, which serve as nodes, and Top-UAVs, which manage the Bottom-UAVs and communicate with the control center. In addition, based on the analyzed delay model, an algorithm was proposed to optimize the packet delay generated by each UAV.

Asheralieva and others [16] proposed an algorithm that optimizes the queuing delay of the BS that processes content requests in cloud-based content delivery networks (CDNs). CDNs include a device-to-device link and a UAV-based BS. The proposed algorithm aims to minimize the content transfer cost and delay.

Li and others [17] proposed an algorithm to minimize the packet delay that occurs when performing multi-hop communication to a macro-cell BS in a multi-layer UAV network. The proposed algorithm analyzes the altitude and queuing model considering the coverage of the UAV in charge of each layer. In addition, an optimization algorithm was proposed to minimize the network packet delay.

Wei and others [18] proposed a three decision-making algorithm to optimize the energy consumption and delay of each task in a cloud-to-thing continuum, where UAVs are used as fog nodes. Based on the task and dataset size of

the mobile device, the UAV finds a suitable spot and then performs communication by calculating the processing frequency and transmission power with a minimized energy consumption and delay.

Zhang and others [19] presented a theory for minimizing the average energy consumption of smart mobile devices and UAVs in UAV-assisted mobile edge computing systems that handle stochastic computation tasks. In [9], an algorithm that analyzes the amount of computation offloading, resource allocation, and flying trajectory scheduling of the UAV and performs joint optimization was proposed.

Koushik and others [20] proposed a channel model based on the individual error rate of the physical layer, data layer, and routing layer in a manned-and-unmanned network, where UAVs are used as relay nodes. In addition, a multi-hop queuing model with M/G/1 preemptive repeat priority (MHQ-PRP) was proposed, which is a queuing model suitable for UAV swarm networks. The paper also proposed a positioning algorithm to find the location of a UAV suitable for communication by using the channel data and queuing model based on deep Q-learning.

Zhang and others [21] proposed a scheme that can classify and process packets into respective networks considering the priority, delay, and resource allocation of packets received by a UAV supporting multiple radio access networks.

2.2 | Apache Spark Streaming

In this study, Spark Streaming is applied to support the computing processes for real-time UAV control and network services, which include edge computing. Spark Streaming was selected because it is one of the most popular real-time processing platforms in the Apache big data series. This is because of its accuracy and reliability, where a mini-batch size of 0.5 s–3 s is most suitable for UAV and AP control. The main differences between Spark and Spark Streaming are the input processing methods used in big data engines. Like its name, Spark Streaming receives input in the form of a stream and divides the stream into small batches (that is, mini-batches) based on time units. In Spark Streaming, a series of uniform-sized mini-batches is called a discretized stream (DStream). Input DStreams are processed in Spark Streaming just like Spark processes batches, but the batch sizes are much smaller in Spark Streaming, and the execution time of one batch processing is also much smaller.

Figure 1 shows the Apache Spark Streaming system. When the input data arrive at the receiver, data are distributed and replicated for two reasons. The distribution of data will enable parallel processing, which is faster, and in case of an error, recovery of the resilient distributed datasets (RDDs) will be more effective and robust [22,23].

When the computation of the current batch is unrelated to the previous batches, the process is called a stateless

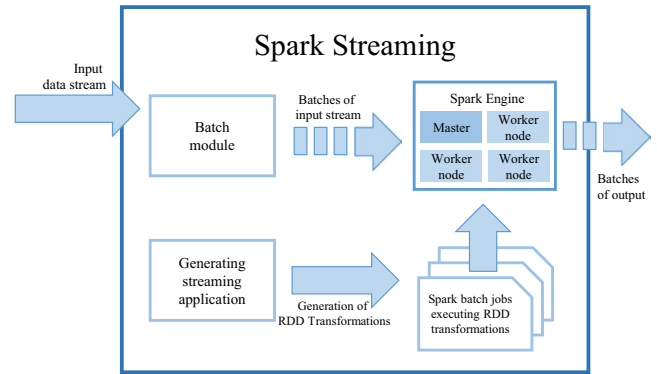


FIGURE 1 Apache Spark Streaming system

transformation. On the other hand, when the previous batches are necessary for the computation of the current batches, the process is called a stateful transformation, where additional techniques are applied in the Spark Streaming process. The most widely used techniques are window-based operations, and a key transformation is `updateStateByKey()`. Such transformations gather several batches into one unit, window the composition of the group, and process the windowed range of the DStream in the Spark Streaming engine. The window slides down the DStream, and the data are sequentially processed continuously.

The Spark Streaming system uses RDDs for fault tolerance, but differentiates itself from Spark by using checkpointing for better recovery of stragglers. All RDDs are stored in memory, and the system additionally replicates some of the RDDs periodically in its external storage (eg, saving every 10th RDD or every 10 s in Hadoop distributed file system, when the input batch size is 1 s). Therefore, when the computation of a certain partition takes an exceptionally long time, it will recall the data from the external storage and recompute in parallel on a different worker node. However, in order to achieve a performance suitable for real-time processing of the UAV-based APs applied in this study, it is important to set an adequate period for checkpointing. When checkpointing occurs in small time intervals, it will degrade the performance because it will allocate too much time to saving the RDDs externally. On the other hand, when checkpointing is conducted at longer periods, the system may not be able to recover from faults and stragglers in real-time. Therefore, a checkpointing period must be set to satisfy the real-time requirements of UAV control and data processing in the APs.

3 | SYSTEM MODEL

The structure of a network formed using hierarchical UAVs uses the UAVs in the HAP as the mobile network's backhaul backbone, as shown in Figure 2. UAVs in the HAP

are located at an altitude of about 17 km–22 km [5] and communicate with the UAV control center, general static cloud, and the UAVs in the low altitude platform (LAP). The UAVs in the HAP operate as an access network and deliver the data to the general static cloud, which is connected to the corresponding HAP. In addition, UAVs in the HAP receive control signals from the UAV control center and adjust their positions as well as the positions of the UAVs in the LAP.

UAVs in the LAP perform the role of the BS, providing direct wireless access to the UEs. Most UAVs in the LAP range are within an altitude of 0.1 km–20 km [5], but a more common range of a UAV in the LAP serving as a BS would be within the altitude range of 0.1 km–5 km, which is considered in this study as well. UAVs in the LAP handle the process of data generated from the UE. In addition, UAVs send data and tasks collected from the UE to the general static cloud through the HAP for processing.

In order to provide seamless networking connections to all UEs, the UAV control center consistently adjusts the positions of the UAVs. The UAV control center communicates with the HAP UAVs and the LAP UAVs to coordinate the

signaling and data flows and maintain optimal positioning of the UAVs.

It is impractical to handle an unlimited number of dataflows owing to the performance or power issues of the UAVs. Therefore, it is essential to control the input data rate of each UE in order to process all the given dataflows. Therefore, in order to maintain the HAP and LAP-based network, it is crucial to know the maximum number of dataflows and input data rate that can be sustained based on the communication and the DStream processing error rates while satisfying a designated time delay bound for real-time control and data services, which is the objective of this study.

Figure 3 shows the communication process in the HAP and LAP. Figure 3A shows the communication links connecting the UEs within a service radius supported by UAVs in the same LAP. In this case, the information communicated among the UEs only needs the support of the LAP using a few UAV hops.

If the communication distance far exceeds the service radius of one LAP, the HAP participates in the communication, as shown in Figure 3B. Because HAPs can enable communication over a longer distance than a LAP, the HAP serves the

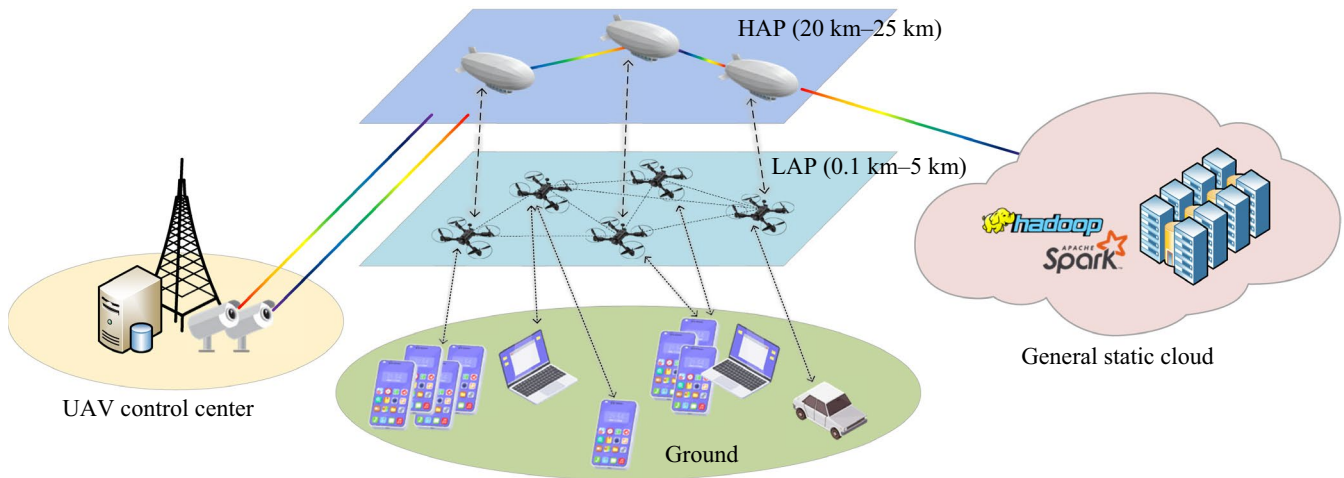


FIGURE 2 Hierarchical UAV network system model

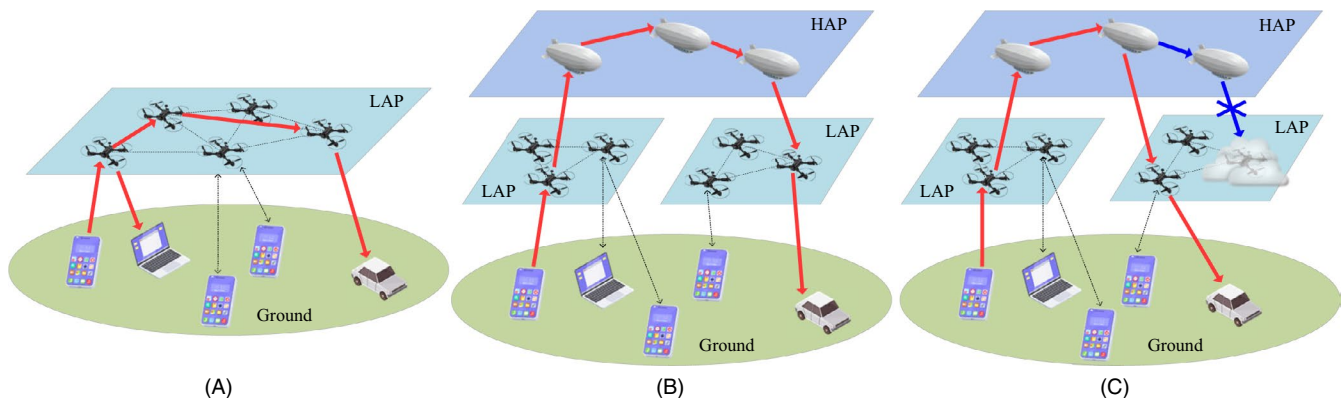


FIGURE 3 Hierarchical UAV network communication model

role of receiving packets from the LAP and transmitting them to another LAP close to the destination UE.

When a LAP UAV close to the destination UE cannot communicate due to clouds or terrain blockage, the LAP will attempt to communicate by searching for an alternative UAV route, as shown in Figure 3C. In this case, the LAP UAV communicating with the destination UE will avoid the obstacle and take on the role of a BS for the UE. If communication with the UE is impossible with the current arrangement of UAVs in the LAP, then the control center will relocate the UAVs in the LAP to establish communication with the UE.

4 | COMBINED TIME BOUND OPTIMIZATION OF UAV CONTROL, COMMUNICATION, AND DATA PROCESSING

Two aspects of time are need to be considered in the proposed system model. One is the time it takes for the control center to adjust the UAV (T_{cc}). The other is the time (T_{DP}) it takes for the general static cloud to receive and process the data. To optimize T_{cc} , the UAV NCS [24] needs the UAVs to apply the computed modifications in order to maintain the quality of the network performance. To optimize T_{DP} , modifications were applied to the Apache Spark Streaming process to satisfy the real-time delay bounds even when errors occur in the communication and data computing process. The SRSFAF scheme proposed in this paper is different from the OptEx [25] and Spark adaptive failure-compensation (SAF) [26] schemes because of how it was designed to cope with errors in the communication and data computing process while satisfying the real-time delay bounds that support control of the UAVs in the HAP and LAP as well as the network data services.

4.1 | UAV NCS

The UAV NCS [24] mainly analyzes the performance of the UAV system based on the cloud environment that processes the DStream. In this scheme, the UAV NCS [24] was modified and used according to the UAV control part of the proposed system model. The UAV NCS needs to manage the UAV control center to adjust the positions of the UAVs and respond to the UAV requests within a predetermined time of T_{cc} . To ensure this, the UAV NCS must be assigned a suitable value of N_1 or λ for a given system environment. The parameters used in the proposed control scheme are listed in Table 1, where $F(\tau)$ is computed based on (1)–(4), which is explained as follows.

Equation (1) calculates the ratio of successful transmissions based on the time-varying position of the UAVs and the reception power of the signals. The ratio is determined by several variables,

including the service radius, transmission power, receiver sensitivity, and velocity of the UAVs, and is based on the equation.

$$\begin{aligned} P_s(v, R) &= \Pr[\text{GR} \geq \beta] \\ &= \iint_{\substack{\left(\frac{0.3ht}{4\pi rf}\right)^2 \text{GT} \geq \beta \\ 0 < r \leq R}} \frac{1}{R} f(h) dh dr \\ &= \iiint_{\substack{x^2 + (y+v)^2 \geq \left(\frac{4\pi rf}{0.3}\right)^2 \frac{\beta}{\text{GT}} \\ 0 < r \leq R}} \frac{1}{2\pi R} e^{-\frac{x^2+y^2}{2}} dx dy dr, \end{aligned} \quad (1)$$

where (x, y) represents the channel fading distribution variables of a LAP UAV, v is the speed of a LAP UAV, R is the service radius of the LAP, r is the distance between the UAV in the LAP and the UE with which it is communicating, GT is the transmission power, and $\text{GR} = (0.3ht/4\pi rf)^2 \text{GT}$ is the receive power.

A set of auxiliary variables is needed in the computation of $F(\tau)$, which can be computed using the probability of successful transmission from (1), number of UAVs, packet arrival rate, and service rate. The definitions of the auxiliary variables are provided below.

$$\begin{aligned} \beta_1 &= P_s(v, R) \mu_g / 2N_1 - \lambda, \\ \beta_2 &= P_s(v, R) \mu_g / 2 - N_1 \lambda, \\ \beta_3 &= \mu_e - N_1 \lambda, \\ \beta_4 &= \mu_p - N_1 \lambda, \\ \beta_5 &= \mu_o - N_1 \lambda, \\ \beta_6 &= \mu_d - \delta N_1 \lambda, \end{aligned} \quad (2)$$

where N_1 is the number of dataflows, which represent the number of links simultaneously connected within the LAP.

Using the auxiliary variables from (2) and [27], $F(\tau)$ can be computed from (4), which uses (3).

$$\begin{aligned} f_n(t) &= \sum_{i=1}^n \beta_i^{k_i} e^{-t\beta_i} \sum_{j=1}^{k_i} \frac{(-1)^{k_i-j}}{(j-1)!} t^{j-1} \\ &\times \sum_{\substack{m_1 + \dots + m_n = k_i - j \\ m_i = 0}} \prod_{\substack{l=1 \\ l \neq i}}^n \binom{k_l + m_l - 1}{m_l} \frac{\beta_l^{k_l}}{(\beta_l - \beta_i)^{k_l + m_l}}, \end{aligned} \quad (3)$$

$$F(T_{CC}) = \Pr[T \leq T_{CC}] = \int_0^{T_{CC}} [\delta f_6(t) + (1 - \delta) f_5(t)] dt. \quad (4)$$

TABLE 1 Parameters for the UAV NCS

Variables	Expressions
$F(\tau)$	Probability that the response time of the whole system is smaller than one control period
β_i	Auxiliary variables of $F(\tau)$, $i = 1, \dots, 6$
R	Service radius of the LAP
β	Receiving sensitivity
GT	Transmitting power
v	Speed of the LAP
h	Channel fading coefficient
r	Distance between a UE and the LAP
f	Frequency of the signal
T_{CC}	Interval of a one-time control period of the UAV
N_1	Number of dataflows controlled by each LAP
λ	Average packet arrival rate
μ_g	Bandwidth of the LAP's output
μ_e	Service rate of the entering server
μ_p	Service rate of the processing server
μ_o	Service rate of the output server
μ_d	Service rate of the database server
δ	Access probability of the database server

In (3), n is used to denote the index number of the auxiliary variables, which has a range of $1 < n < 6$, and k_i denotes the number of components based on the same parameter β_i .

4.2 | SRSFAF

The method of how the proposed SRSFAF scheme computes T_{DP} is described in this section. Time T_{DP} is the time required to process the data received through the UAV network. SRSFAF was designed by extending the OptEx model [25] and SAF model [26] by making modifications to estimate the performance of Spark Streaming supported by the HAP and LAP. The proposed SRSFAF scheme focuses on optimal resource allocations to satisfy real-time constraints, even when errors occur in the communication and DStream processes. Therefore, extensions to the variable sharing phase and the computation phase of the OptEx model [25] and SAF model [26] were applied. The parameters used in the SRSFAF scheme are presented in Table 2.

Estimation of the time to process a batch file in SRSFAF is as follows.

$$T_{DP} = T_{vs} + T_{comp} + T_{add} = T_{vs} + T_{commn} + T_{exec} + T_{add}. \quad (5)$$

In (5), T_{add} refers to the additional time needed due to an error that may occur during the job execution, and T_{add} consists of two parts, as shown below:

TABLE 2 Parameters for SRSFAF

Variables	Expressions
T_{DP}	Estimation time of the data processing phase
T_{vs}	Variable sharing time
T_{comp}	Computation time
T_{commn}	Communication time
T_{exec}	Execution time
T_{add}	Time added because of an error
$P_{e_{commn}}$	Error rate in the communication phase
$P_{e_{exec}}$	Error rate in the execution phase
$T_{e_{vs}}$	Variable sharing time in the error recovery phase
$T_{e_{commn}}$	Communication time in the error recovery phase
$T_{e_{exec}}$	Execution time in the error recovery phase
M_d	Processing time of a batch file
s	Input data size
i	Number of iterations
n	Number of worker nodes
θ	Scaling factor for the baseline time

$$T_{add} = P_{e_{commn}} T_{commn} + P_{e_{exec}} (T_{e_{vs}} + T_{e_{commn}} + T_{e_{exec}}), \quad (6)$$

where $P_{e_{commn}} T_{commn}$ represents the extra time due to an error that might occur during the communication process. In this case, only T_{commn} is added because the batch file should be retransmitted from the receiver. The term $P_{e_{exec}} (T_{e_{vs}} + T_{e_{commn}} + T_{e_{exec}})$ represents the time added due to an error that occurred during execution. A worker node that has an error during execution goes into a process to recover from the error. At this time, the batch file assigned to the failed node cannot be processed. Therefore, the driver distributes the batch file to other worker nodes for processing. In this process, the DStream variable sharing and communication phase tasks are performed once more. Based on the OptEx model [25] and (6), T_{DP} is expressed as.

$$\begin{aligned} T_{DP} &= niC + \frac{As}{n} + \frac{iB}{n} + P_{e_{commn}} \frac{As}{n} \\ &\quad + P_{e_{exec}} \left\{ (n-1)iC + \frac{A \frac{s}{n}}{n-1} + \frac{i \frac{B}{n}}{n-1} \right\} \quad (7) \\ &= niC + \frac{As}{n} + \frac{iB}{n} \\ &\quad + P_{e_{commn}} \frac{As}{n} + P_{e_{exec}} \left\{ (n-1)iC + \frac{As}{n(n-1)} + \frac{iB}{n(n-1)} \right\}, \end{aligned}$$

where $A = \theta_{commn} T_{commn}^{baseline}$, $B = M_d$, and $C = \theta_{vs} T_{vs}^{baseline}$, where θ_{commn} and θ_{vs} are scaling factors that have values between 0 and 1. $T_{commn}^{baseline}$ and $T_{vs}^{baseline}$ are the average times taken to perform the process (commn, vs) in the big data system. Assuming that n is sufficiently large, T_{DP} can be transformed into.

$$\begin{aligned}
T_{DP} &\cong niC + \frac{As}{n} + \frac{iB}{n} + P_{e_{\text{commn}}} \frac{As}{n} \\
&\quad + P_{e_{\text{exec}}} \left\{ niC + \frac{As}{n^2} + \frac{iB}{n^2} \right\} \\
&= niC(1 + P_{e_{\text{exec}}}) + \frac{1}{n} (iB + As(1 + P_{e_{\text{commn}}})) \\
&\quad + \frac{1}{n^2} (P_{e_{\text{exec}}} (As + iB)) \\
&= \alpha n + \beta \frac{1}{n} + \gamma \frac{1}{n^2},
\end{aligned} \tag{8}$$

where $\alpha = iC(1 + P_{e_{\text{exec}}})$, $\beta = iB + As(1 + P_{e_{\text{commn}}})$, and $\gamma = P_{e_{\text{exec}}} (As + iB)$.

The number of worker nodes (n) required to satisfy the given time-out T_{object} can be obtained through (9) and (10).

$$T_{DP} \cong \alpha n + \beta \frac{1}{n} + \gamma \frac{1}{n^2} \leq T_{\text{object}}, \tag{9}$$

$$\alpha n^3 - T_{\text{object}} n^2 + \beta n + \gamma \leq 0. \tag{10}$$

For variable n , the roots that satisfy (10) are obtained as (11)–(13).

$$\begin{aligned}
n_1 &= \frac{T_{\text{object}}}{3\alpha} - \frac{1}{3\alpha} \sqrt[3]{\frac{\Delta + \sqrt{\Delta^2 - 4(T_{\text{object}}^2 - 3\alpha\beta)^3}}{2}} \\
&\quad - \frac{1}{3\alpha} \sqrt[3]{\frac{\Delta - \sqrt{\Delta^2 - 4(T_{\text{object}}^2 - 3\alpha\beta)^3}}{2}},
\end{aligned} \tag{11}$$

$$\begin{aligned}
n_2 &= \frac{T_{\text{object}}}{3\alpha} + \frac{1+i\sqrt{3}}{6\alpha} \sqrt[3]{\frac{\Delta + \sqrt{\Delta^2 - 4(T_{\text{object}}^2 - 3\alpha\beta)^3}}{2}} \\
&\quad + \frac{1-i\sqrt{3}}{6\alpha} \sqrt[3]{\frac{\Delta - \sqrt{\Delta^2 - 4(T_{\text{object}}^2 - 3\alpha\beta)^3}}{2}},
\end{aligned} \tag{12}$$

$$\begin{aligned}
n_3 &= \frac{T_{\text{object}}}{3\alpha} + \frac{1-i\sqrt{3}}{6\alpha} \sqrt[3]{\frac{\Delta + \sqrt{\Delta^2 - 4(T_{\text{object}}^2 - 3\alpha\beta)^3}}{2}} \\
&\quad + \frac{1+i\sqrt{3}}{6\alpha} \sqrt[3]{\frac{\Delta - \sqrt{\Delta^2 - 4(T_{\text{object}}^2 - 3\alpha\beta)^3}}{2}}.
\end{aligned} \tag{13}$$

In (11)–(13), $\Delta = -2T_{\text{object}}^3 + 9\alpha\beta T_{\text{object}} + 27\alpha^2\gamma$. Because n is the number of worker nodes, the smallest n becomes the optimal number of worker nodes (N_2) to be applied.

$$N_2 = \{ \lceil \min(n_i) \rceil | n_i > 0, n_i \in \mathbb{R}, i = 1, 2, 3 \}. \tag{14}$$

In (14), $\lceil x \rceil$ represents the least integer greater than or equal to x .

5 | CONTROL FLOWCHART FOR UAV AERIAL NETWORK

Within a given time period, the number of dataflows (N_1) that one UAV can handle or the number of optimal worker nodes (N_2) necessary to deal with the aggregated input data rate (λ) can be calculated based on the control process presented in Figure 4. Among the inputs in Figure 4, ρ is the guaranteed minimum rate of $F(\tau)$, which indicates the ratio of packets that are successfully processed within a time period (τ) during the communications between the UAVs. The performance bound for $F(\tau)$ can be defined based on ρ , which is a user-specified value according to the required quality of service (QoS) of the network.

The control center and HAP manage the QoS of the network by calculating changes and updating $F(\tau)$, whenever a change in the network environment occurs using the scheme presented in Section 4. The control center and HAP are involved from the start of the control system to the part that derives the optimal N_1 or λ . The relevant part is as follows. When the system starts, it initializes counter (x_N, x_λ) first. The counter adjusts the λ and N_1 values so that they do not exceeding a certain range when they do not match the set range. Afterward, if there is a change in the network environment, the input data values are updated. The user can select one control parameter from among λ and N_1 , and the control system computes and specifies the selected control parameters that satisfy $\rho \leq F(\tau)$ based on (1)–(4).

Parameters ψ_N and ψ_λ , which are values between 0 and 1, respectively, represent the ratio of how much N_1 and λ are to be reduced when the control parameter value that satisfies $\rho \leq F(\tau)$ cannot be found. When a proper λ value cannot be found in the calculation process, the value of N_1 is reduced by $(1 - \psi_N)$. Then, the value of λ that satisfies $\rho \leq F(\tau)$ is calculated again. When an appropriate N_1 value is not found in the calculation process, the value of λ is reduced by $(1 - \psi_\lambda)$. Then, the value of N_1 that satisfies $\rho \leq F(\tau)$ is calculated again. If the control parameter values that satisfy the requirements cannot be found even after this process, it is repeated again, where (x_N, x_λ) is used as a counter value to keep track of the number of times this process is repeated.

When the values of ψ_N and ψ_λ are larger, the control parameter that satisfies the specified probability can be found more quickly. However, if ψ_N and ψ_λ are too large, opposing problems will occur. For example, if N_1 is reduced by a large margin, the LAP may need to deploy more UAVs than necessary, and reducing λ too much will result in a significant reduction in the transmission rate; thus, parameter control needs to be carefully conducted.

After calculating the optimal N_1 or λ , the control center and HAP apply the results to the LAP, where N_1 adjusts the dataflow managed by adding or reducing the number of UAVs in the LAP. In addition, λ changes the value set in the currently deployed UAVs of the LAP to adjust the network speed within the service range managed by the LAP.

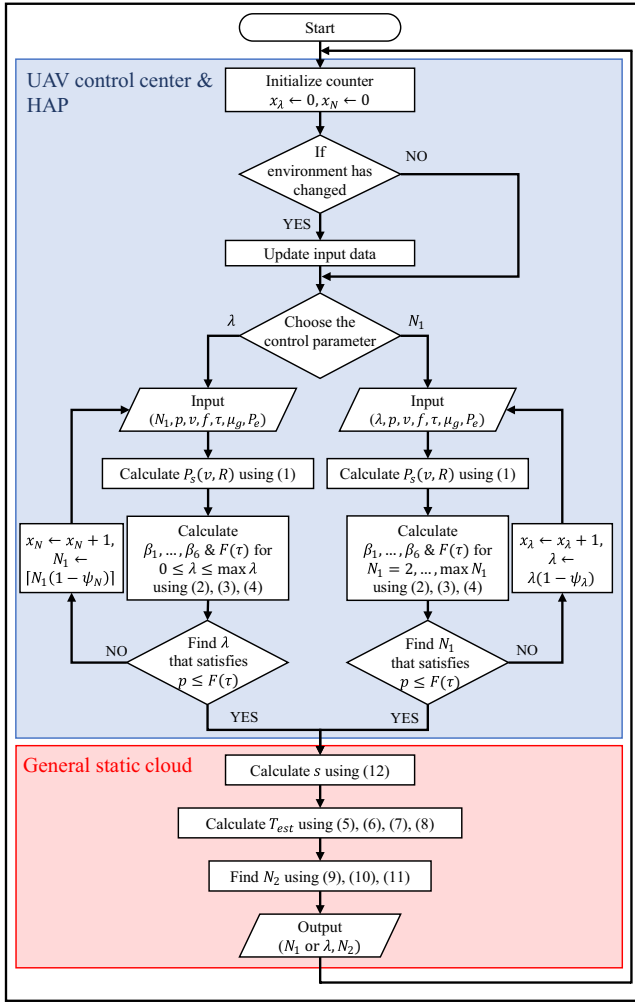


FIGURE 4 UAV network control flowchart

Changes in N_1 and λ made through the above process result in changes that need to be made in N_2 , which is adapted to ensure that the Spark DStream process can be completed in real time.

When N_1 or λ , based on (1)–(4), seems relevant, the input size (s) can be evaluated with the given input parameters in the form of.

$$s = N_1 \times n \times \lambda \times t \times \omega, \quad (15)$$

where n denotes the overall number of UEs of which one hierarchical UAV network takes charge, t is a system-specified batch time in the Spark DStream process, and ω is the input size in bytes per packet. Using s from (15) and (5)–(8), T_{DP} is obtained. Then, the process compares T_{DP} and the object time as in (9) and (10), and the number of worker nodes (N_2) necessary for real-time DStream processing can be calculated.

If there is no requested command, this algorithm can be performed again periodically to maintain network stability, or

it could be used on demand when any changes are made to the HAP and LAP networks.

6 | SIMULATION RESULT

6.1 | Simulation environment

In the performance analysis, it is assumed that the initial preparation for DStream has been adequately set up with sufficient memory, where errors in the communication and data processing may occur. For the simulation experiments, assumptions on several experimental parameters were made as follows. It is presumed that every communication proceeds with either a radio frequency or an FSO signal, and the frequency channel is in the 3 GHz band. The packet size was fixed at 1500 bytes per packet and the service radius of a UAV was set to 3 km. It is assumed that the transmitting power (GT) was 1 W and the receiver sensitivity (β) was 5 mW. UAVs in the LAP moved at an average speed (v) of 1 m/s, and the value of τ was set to 20 ms. In the DStream, a batch size was chosen according to the object time limit. Possible candidates for the DStream batch size ranged from 0.5 s–5 s in increments of 0.5 s.

6.2 | Simulation results

Changes in the communication success rate ($P_s(v, R)$) according to the movement of the UAVs in the LAP (v) are shown in Figure 5. The graph shows that $P_s(v, R)$ decreases as the speed of the UAVs in the LAP increases. In particular, when the service radius is small, $P_s(v, R)$ decreases faster. The larger the service radius is, the higher $P_s(v, R)$ becomes. It can be considered that a larger service radius leads to more stable communication. However, a wide service radius means that the power consumption of the UAVs and number of dataflows to be managed on each UAV will increase. Therefore, each UAV's communication success rate may increase, but the overall service performance may deteriorate.

Changes in the number of UEs (n) required within the service radius are in connection with parameter N_1 , as presented in Figure 6. When N_1 is the same, a smaller λ results in a greater $F(\tau)$, but as N_1 increases, the value of $F(\tau)$ decreases, and as the value of λ increases, $F(\tau)$ decreases. The control center and the HAP may act by adding a UE within the service range if $F(\tau)$ is expected to be smaller than a specified threshold. By adding a UAV in the LAP, the number of N_1 can be reduced to maintain $F(\tau)$ above a certain level. If $F(\tau)$ cannot be increased above the threshold even if N_1 is decreased, then $F(\tau)$ is increased by decreasing the value of λ .

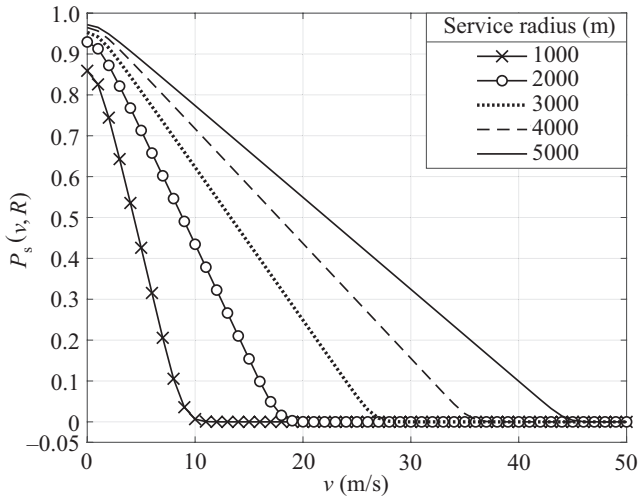


FIGURE 5 Communication success rate with v

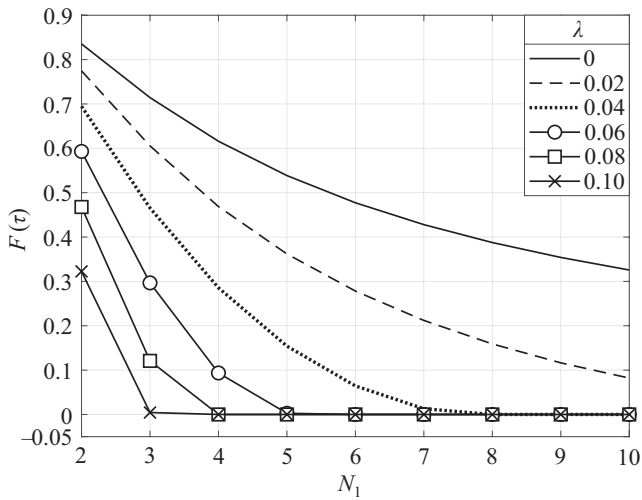


FIGURE 6 Time delay probability with N_1

Figures 7 and 8 illustrate the performance distribution of the time delay probability $F(\tau)$, according to λ and N_1 based on T_{CC} . The horizontal plane in the middle represents the minimum guaranteed ratio p , which was set to $p = 0.4$ in the simulation. It can be seen that $F(\tau)$ is largest when λ and N_1 are the smallest, and $F(\tau)$ decreases as λ and N_1 increase. The range of λ and N_1 that satisfy p is easily identifiable in the graph as the curved part of $F(\tau)$ above the $p = 0.4$ plane.

Figures 7 and 8 are based on $\mu_g = 1$ and $\mu_g = 0.75$, respectively. By comparing Figures 7 and 8, it is evident that the peak value of $F(\tau)$ decreases as μ_g decreases. Likewise, for an equivalent value of p , the range of λ and N_1 satisfying p also diminishes for a smaller μ_g value.

Figure 9 shows the difference in object time based on the error rate. The error rates ($P_{e_{exec}}, P_{e_{comm}}$) in the simulation were set to 0.02, 0.05, and 0.08. In each case, $P_{e_{exec}}$ and $P_{e_{comm}}$ were assumed to be the same. When comparing the performance

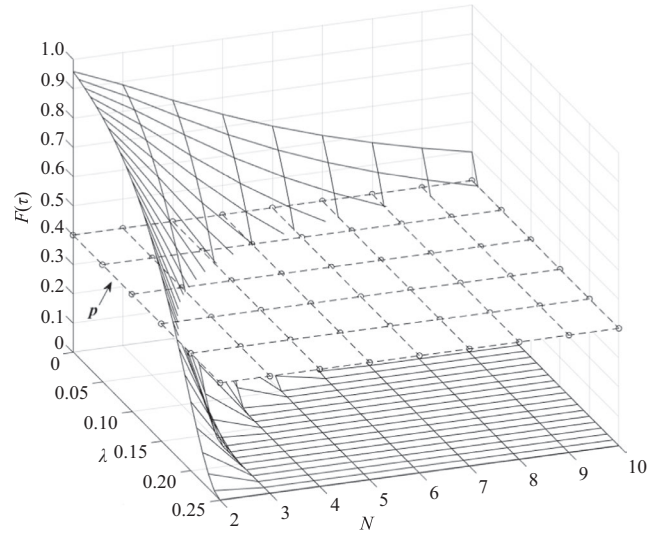


FIGURE 7 Time delay probability with λ and N_1 ($\mu_g = 1$)

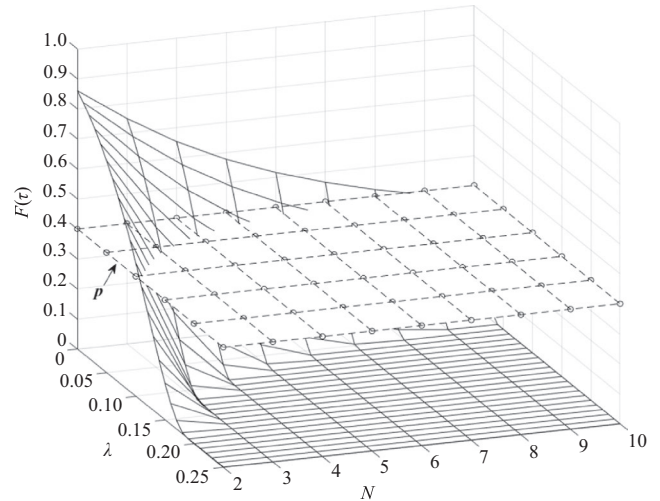


FIGURE 8 Time delay probability with λ and N_1 ($\mu_g = 0.75$)

based on the error rate, the difference in performance with OptEx is based on the fact that OptEx maintains the same number of worker nodes, whereas SRSAF adjusts its number of worker nodes to an optimal number based on the error rate experienced in the communication and DStream. Since SRSAF considers the extra time required to recover from errors, it utilizes more worker nodes than OptEx. As shown in Figure 9, OptEx exceeds the object time in every case, and this becomes worse when the input data size increases with an increased batch size.

In the case of SAF, it slightly exceeds the object time. This is because SAF only considers the additional resources for the variable sharing time and communication time due to errors in the processing of an RDD. However, it does not allocate additional resources to compensate for errors occurring during communication between worker nodes and the time to

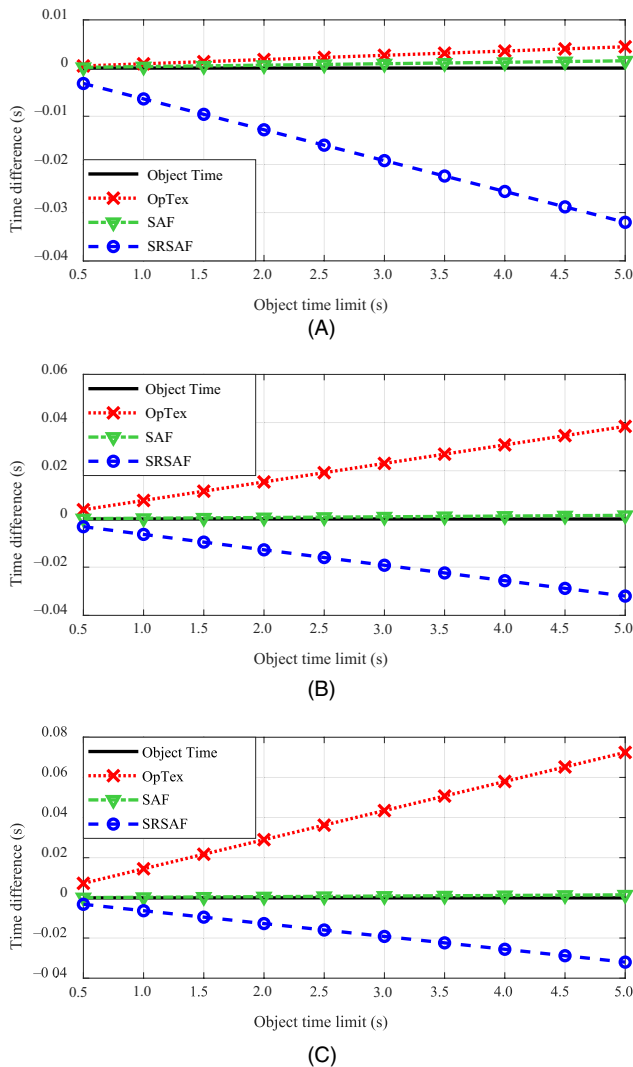


FIGURE 9 Time difference with object time: (A) $P_{e_{exc}} = P_{e_{commn}} = 0.02$, (B) $P_{e_{exc}} = P_{e_{commn}} = 0.05$, and (C) $P_{e_{exc}} = P_{e_{commn}} = 0.08$

process the remaining RDD after an error occurs during RDD processing. This causes a time delay in the elapsed processing time. Therefore, SAF estimates the number of necessary worker nodes due to processing errors, which may cause the object time to be exceeded.

In SRSAF, not only the processing time for error recovery in DStream but also the additional time for communication error recovery between worker nodes are considered. In Figure 9, the SRSAF does not exceed the object time for any of the cases tested. Moreover, even when the batch size increases because SRSAF re-computes the number of worker nodes based on an optimized estimation of resource allocation, the scheme safely reserves a gap between the object time and real processing time, making the system more robust against sudden error events.

Figure 10 shows the relationship between N_1 and the DStream processing time. The object time to process

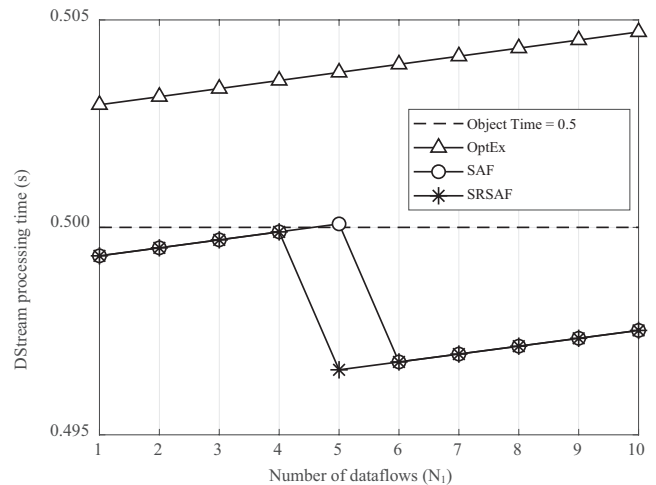


FIGURE 10 DStream processing time based on N_1

DStream is set to 0.5 s, which is the smallest value that can be used in Spark streaming big data systems. In the case of OptEx, the object time is not satisfied for any value of N_1 . In the case of SAF, all cases can be processed without exceeding the object time, except for $N_1 = 5$. SRSAF does not exceed 0.5 s in any of the N_1 cases tested. These adaptations were caused by increasing the number of worker nodes (N_2), in which SAF and SRSAF predict that the processing time will exceed the object time due to an error. Figure 10 shows that the proposed SRSAF scheme can provide a smaller DStream processing time with $N_1 = 5$, than when SAF uses $N_1 = 6$. Based on Figure 10, it can be concluded that SRSAF is more effective than SAF in the range of interest.

Figure 11 shows the relation between ψ and the elapsed time. The initial value of λ is 0.25. As x_λ increases by 1, λ decreases with a proportion of $(1 - \psi)$, where ψ is set to the three cases of 0.1, 0.2, and 0.3. The system error rate is fixed to 0.05 and the object time is set to 0.5 s.

In the case of OptEx, the elapsed time decreases when x_λ increases. This is because the input data size also decreases when λ decreases. As x_λ increases, λ converges to a certain value. However, in almost every case, regardless of how large x_λ is set, OptEx exceeds the performance limit.

On the other hand, unlike OptEx, there are some incline transitions present in the performance curves of SAF and SRSAF. This happens because as the input data size decreases, SAF and SRSAF select a smaller number of worker nodes based on the optimal number (to satisfy the object time) derived. Therefore, the part where the elapsed time increases (in the performance curves of SRSAF in Figure 11) indicates a reduction in the number of the worker nodes.

SAF has a smaller estimation time than SRSAF because it considers only the error time that occurs in RDD processing.

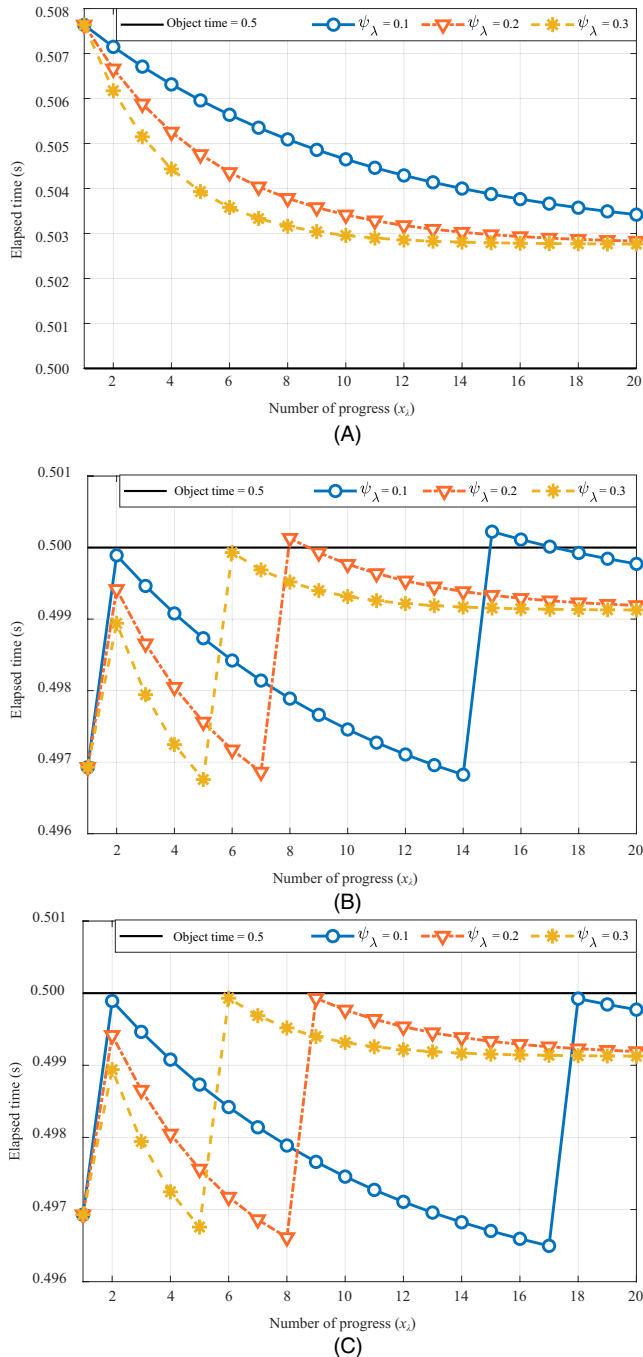


FIGURE 11 Elapsed time performance based on ψ : (A) OptEx, (B) SAF, and (C) SRSFAF

Therefore, the number of optimal worker nodes changes at a lower x_λ point, and the elapsed time increases. However, in SAF, the object time is exceeded at the point where the elapsed time increases. This shows that SAF cannot sufficiently predict the additional time due to errors occurring during processing.

In contrast, SRSFAF succeeds in processing within the time bound in all three cases. In the same manner, when x_λ increases, both λ and the input data size decrease, lessening the elapsed time.


7 | CONCLUSION

In this paper, the SRSFAF scheme was proposed to control the hierarchical UAV-based HAP and LAP network communication and DStream process in real time based on the time bound set for the control operation and data services. In a network environment where the UAVs in the HAP serve as a backhaul backbone network and the UAVs in the LAP serve as the BSs, the proposed SRSFAF scheme makes it possible to obtain the optimal number of dataflows and input data rates that can be processed within the real-time operation bounds. Based on the optimum input data rate, the proposed scheme computes the optimal number of worker nodes required considering the error rates. The results show that the SRSFAF scheme can provide a reliable performance that satisfies the required performance bounds for the parameter range of interest and exceeds the performance of the OptEx and SAF schemes.

ORCID

Seungwoo Seo  <https://orcid.org/0000-0003-0242-4336>

Da-Eun Ko  <https://orcid.org/0000-0002-3253-2429>

Jong-Moon Chung  <https://orcid.org/0000-0002-1652-6635>

REFERENCES

1. M. Sharma, D. Chadha, and V. Chandra, *High-altitude platform for free-space optical communication: performance evaluation and reliability analysis*, IEEE/OSA J. Opt. Commun. Netw. **8** (2016), 600–609.
2. M. Mozaffari et al., *A tutorial on uavs for wireless networks: applications, challenges, and open problems*, IEEE Commun. Surveys Tutorials **21** (2019), 2334–2360.
3. Z. Zhang et al., *6G wireless networks: vision, requirements, architecture, and key technologies*, IEEE Veh. Technol. Mag. **14** (2019), 28–41.
4. M. A. Esmail, H. Fathallah, and M. Alouini, *Channel modeling and performance evaluation of FSO communication systems in fog*, In Proc. int. Conf. Telecommun. (Thessaloniki, Greece), 2016, pp. 1–5.
5. S. Song et al., *Analysis of wireless backhaul networks based on aerial platform technology for 6g systems*, CMC Comput. Materials Continua. **62** (2020), 473–494.
6. S. Chandrasekharan et al., *Designing and implementing future aerial communication networks*, IEEE Commun. Mag. **54** (2016), 26–34.
7. N. Wang, E. Hossain, and V. K. Bhargava, *Backhauling 5G small cells: a radio resource management perspective*, IEEE Wireless Commun. **22** (2015), 41–49.

8. S. J. Lee et al., *Development of autonomous flight control system for 50m unmanned airship*, in Proc. Intell. Sens., Sens. Netw. Inf. Process. Conf. (Melbourne, Australia), 2004, pp. 457–461.
9. R. La Scalea et al., *Opportunities for autonomous UAV in harsh environments*, in Proc. Int. Symp. Wireless Commun. Syst. (Oulu, Finland), 2019, pp. 227–232.
10. M. A. Esmail, H. Fathallah, and M. S. Alouini, *Outdoor FSO communications under fog: Attenuation modeling and performance evaluation*, IEEE Photon. J. **8** (2016), 1–22.
11. S. Wan et al., *Towards big data processing in IoT: Path planning and resource management of UAV base stations in mobile-edge computing system*, IEEE Internet Things J. **7** (2020), 5995–6009.
12. J. Zhang, Y. Zeng, and R. Zhang, *UAV-enabled radio access network: Multi-mode communication and trajectory design*, IEEE Trans. Signal Process. **66** (2018), 5269–5284.
13. G. Wang, L. Zhang, and W. Xu, *What can we learn from four years of data center hardware failures?*, in Proc. Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (Denver, CO, USA), 2017, pp. 25–36.
14. S. Kavulya, et al., *An analysis of traces from a production map-reduce cluster*, in Proc. IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput. (Melbourne, Australia), 2010, pp. 94–103.
15. Q. Zhang et al., *Response delay optimization in mobile edge computing enabled UAV swarm*, IEEE Trans. Veh. Technol. **69** (2020), 3280–3295.
16. A. Asheralieva, and D. Niyato, *Game theory and lyapunov optimization for cloud-based content delivery networks with device-to-device and UAV-enabled caching*, IEEE Trans. Veh. Technol. **68** (2019), 10094–10110.
17. J. Li, and Y. Han, *A traffic service scheme for delay minimization in multi-layer UAV networks*, IEEE Trans. Veh. Technol. **67** (2018), 5500–5504.
18. X. Wei et al., *Joint optimization of energy consumption and delay in cloud-to-thing continuum*, IEEE Internet Things J. **6** (2019), 2325–2337.
19. J. Zhang et al., *Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing*, IEEE Internet Things J. **6** (2018), 3688–3699.
20. M. Koushik, and F. Hu, S. Kumar, *Deep Q-learning-based node positioning for throughput-optimal communications in dynamic UAV swarm network*, IEEE Trans. Cognitive Commun. Netw. **5** (2019), 554–566.
21. Z. Zhang et al., *UAV-enabled multiple traffic backhaul based on multiple RNAs: A batch-arrival-queueing-inspired approach*, IEEE Access **7** (2019), 161437–161448.
22. M. Zaharia et al., *Discretized streams: Fault-tolerant streaming computation at scale*, in Proc. ACM Symp. Oper. Syst. Principles (Farmington, PA), 2013, pp. 423–438.
23. W. Li et al., *Wide-area spark streaming: Automated routing and batch sizing*, IEEE Trans. Parallel Distrib. Syst. **30** (2018), 1434–1448.
24. F. Luo et al., *Stability of cloud-based UAV systems supporting big data acquisition and processing*, IEEE Trans. Cloud Comput. **7** (2019), 866–877.
25. S. Sidhanta, W. Golab, and S. Mukhopadhyay, *Deadline-aware cost optimization for spark*, IEEE Trans. Big Data (2019), <https://doi.org/10.1109/TBDDATA.2019.2908188>.
26. J. Lee, B. Kim, and J. M. Chung, *Time estimation and resource minimization scheme for apache spark and hadoop big data systems with failures*, IEEE Access **7** (2019), 9658–9666.
27. M. Akkouchi, *On the convolution of exponential distributions*, J. Chungcheong Math. Soc. **21** (2008), 501–510.

AUTHOR BIOGRAPHIES



Seungwoo Seo received his BS degree from the School of Electrical and Electronic Engineering, Yonsei University, Rep. of Korea, in 2014. He is currently pursuing a combined MS and PhD degree in electrical and electronic engineering at Yonsei University, Seoul, Rep. of Korea, where he is also a research member of the Communications and Networking Laboratory. His research interests include big data, augmented reality, 6G UAV networks, and trust network security management technology.



Da-Eun Ko received her BS degree in electrical and electronic engineering and BS degree in nano science and engineering from Yonsei University, Seoul, Rep. of Korea, in 2020, where she graduated receiving the highest honor-roll award. She is currently pursuing an MS degree in electrical and electronic engineering at Yonsei University, where she is a researcher at the Communications and Networking Laboratory. Her current research interests include big data, augmented reality, and 6G UAV aerial networks.



Jong-Moon Chung received his BS and MS degrees in electronic engineering from Yonsei University, Seoul, Rep. of Korea, and PhD in electrical engineering from Pennsylvania State University, University Park, Pennsylvania, USA. Since 2005, he has been a professor at the School of Electrical and Electronic Engineering at Yonsei University, where he is currently an associate dean of the College of Engineering and professor in the Department of Emergency Medicine at Yonsei University College of Medicine. From 1997 to 1999, he was an assistant professor and instructor at Pennsylvania State University. From 2000 to 2005, he was at Oklahoma State University (OSU), Stillwater, Oklahoma, USA as a tenured associate professor. Currently he serves as vice president of the IEEE Consumer Electronics Society, editor of the *IEEE Transactions on Vehicular Technology*, associate editor of the *IEEE Transactions on Consumer Electronics*, section editor of the *Wiley ETRI Journal*, and co-editor-in-chief of the *KSII Transactions on Internet and Information Systems*. Dr. Chung will serve as the general chair of IEEE ICCE 2022 and IEEE ICCE-Asia 2020, and was the general chair of several conferences, including IEEE MWSCAS 2011.