Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.058512

ARTICLE

Content Caching Algorithms in Drone-Aided Ad Hoc Networks

Yong Beom Park, Jian Kim, BeomKyu Suh, Ismatov Akobir and Ki-Il Kim*

Department of Computer Science and Engineering, Chungnam National University, Daejeon, 34134, Republic of Korea

*Corresponding Author: Ki-Il Kim. Email: kikim@cnu.ac.kr

Received: 14 September 2024 Accepted: 21 November 2024 Published: 19 December 2024

ABSTRACT

Content delivery networks (CDNs) lead to fast content distribution through content caching at specific CDN servers near end users. However, existing CDNs based on infrastructure cannot be employed in special cases, such as military operations. Thus, a temporary CDN without an existing infrastructure is required. To achieve this goal, we introduce a new CDN for drone-aided ad hoc networks, whereby multiple drones form ad hoc networks and quickly store specific content according to new caching algorithms. Unlike the typical CDN server, the content-caching algorithm in the proposed architecture considers the limited storage capacity of the drone. We present three content distribution algorithms that consider the constraints and mobility of drones. The main contribution of content caching for drone-aided ad hoc networks is to keep partial segments rather than whole content as well as move the drone near to area with a high volume of requests. The proposed scheme is evaluated to demonstrate its feasibility in terms of content acquisition time and utilization in several practical scenarios through simulations. Consequently, acquisition time in CDN to support drone movement is improved by approximately 50% and 40% rather than one in the proposed naive greedy approach as a function of content request interval and size, respectively.

KEYWORDS

Content delivery networks; content caching; unmanned aerial networks; Ad hoc networks; partial segment; duplicated contents; drone

1 Introduction

As the demand for content continually increases, CDNs contribute to reducing users' content acquisition time by locating the content near them. Typically, the content management subsystem in a CDN determines where to place the content. Because the content placement algorithm is vital in determining the acquisition time, it has attracted the interest of many researchers. This effort has been presented in several comprehensive survey papers that address content placement schemes in CDN. Specifically, they include general research trends, technical issues, and challenges. Sahoo et al. [1] presented a method for placing a replica server in a CDN, considering not only deployment, delivery, and update as cost functions but also network properties represented by topology, network performance metrics, and available bandwidth. Using well-known theoretical models for server placement, existing replica server placement algorithms can be classified into the categories of Quality of Service (QoS), consistency, and energy awareness. Two research groups focused

on content placement from the aspect of network infrastructure. Jia et al. [2] addressed collaboration for content delivery and network infrastructure using evolutionary and revolutionary solutions. The former addresses typical content-delivery solutions, whereas the latter addresses emerging content-delivery solutions. Salahuddin et al. presented a literature review on cloud-based CDN (CCDN) [3]. In this survey, specific design criteria are derived and discussed for a content placement algorithm in CCDN. In this survey, access patterns and content popularity are selected for classification.

In addition to these surveys, recent CDN research has focused on two research goals: integration with edge computing and employing machine learning for content placement. Gao et al. [4] presented a new infrastructure by building a small-scale cloud infrastructure at the network edge through centralized and distributed algorithms for the video content placement problem in Multi-access Edge Computing (MEC). Li et al. presented a specific model for smart cities, content caching, and user-association policy [5]. They focused on the impact of user mobility in MEC by introducing long-time scale content caching and short-time scale user association. In MEC, frequent content sharing incurs operational costs in terms of bandwidth and energy consumption, necessitating a balance of tradeoffs. Lin et al. [6] proposed content caching and a migration scheme considering user preferences, thereby developing cost-efficient edge networks. Machine learning-based approaches for CDN have been steadily proposed. In a comprehensive survey, Nomikos et al. [7] presented reinforcement learning (RL)-aided mobile edge caching solutions, classifying them in terms of networking architecture and optimization objectives. As a result, the authors concluded that reinforcement learning-aided caching schemes cause varying complexities nonetheless outperforming conventional approaches.

A more general discussion of ML in CDN was presented in [8], where the authors review and discuss state-of-the-art Machine Learning (ML)-based approaches, thereby concluding that classification/prediction capabilities of ML-based schemes improve network scheduling and parameter adaptation. In addition to a survey paper, Sadeghi et al. [9] presented a new deep Q-network to handle large state spaces in a scalable manner. This approach introduces a new Q-function and optimal caching policy for the parent and leaf nodes. Another RL-based approach was proposed in [10], the primary technical contribution of which is the development of a new algorithm to determine the admission of the requested content by extending a large set of features. Performance evaluation revealed that training in one place can work well for other locations in the same geographic region for diverse traffic classes.

As described above, many studies have been conducted to reduce content acquisition time by locating the content near users. However, these approaches can only be implemented over existing infrastructure; therefore, they cannot be applied to ad hoc networks, where nodes form self-organizing networks. In contrast to infrastructure-based schemes, these approaches build networks and apply content caching. Among the diverse types of ad hoc networks, content delivery networks over vehicular ad hoc networks (VANET) have attracted the interest of several researchers. Research challenges and open issues were addressed in [11], where the authors proposed named data networking (NDN) for VANET and content caching in addition to providing an overview of vehicular edge caching and machine learning-based approaches. Both studies highlight open issues and research challenges. Accordingly, a specific space-efficient caching algorithm was proposed in [12]. Theoretical and analytical models were defined and derived to increase cache space utilization. In [13], content delivery on vehicular clouds (CDVC) was proposed, utilizing the cloud and a named service under a hierarchical architecture, to enable rapid retrieval of content and request aggregation among vehicles. In these studies, the operational environments were assumed to cooperate with the existing infrastructure. Thus, they cannot be applicable to pure ad hoc networks, where no infrastructure is assumed and prompt network construction is required. Representative examples include content sharing during

military operations and temporarily crowded areas. In addition, the models and applications for UAV (Unmanned Aerial Vehicle) caching algorithms in 6G networks were well analyzed in [14].

To satisfy the aforementioned requirements, drone-aided flying ad hoc networks (FANET) were investigated. This type of network has the benefit of low deployment cost in terms of time and device. However, to the best of our knowledge, there is only one study [15] that defines drone-aided content delivery networks, presenting a system model and architecture to request routing and queuing for the blocked request probabilities of drones. The main goal of this study was to distribute the load and reduce latency by employing drones as aerial content delivery points. To achieve this, they introduced different theoretical queuing models for drones and base stations to determine whether loads can be transferred to the drone. In this way, a drone can cover several base stations for load balancing. The authors demonstrated that the proposed drone-aided mobile CDN outperforms the original mobile CDN. However, this architecture has not been fully deployed in drone-aided ad hoc networks. Therefore, its application is limited to the introduction of drones for content delivery in mobile CDN networks. This implies that this approach cannot be deployed without infrastructure, as indicated in previous studies.

To overcome the deployment issues of previous studies, in this paper, we present a new content delivery network using drones. Unlike the typical servers and vehicles in previous approaches, the new algorithm for content placement considers the limitations of computing and storage capacity of drones. A list of acronyms is given in Table 1. The main contributions of this study are as follows:

- **Design of system model:** To enable the practical deployment of the proposed solution, a system model was adopted for drone-aided ad hoc networks. This study is the first to describe a system model and procedure for content placement.
- **Content placement:** The proposed scheme considers segmented content and automatic content aging to overcome the storage limitations of each drone using algorithms with features different from those in previous work.
- **Dynamic mobility:** Unlike the previous CDN approach, a dynamic topology is built for CDN by moving some drones toward specific areas. This reduces acquisition time by forming clusters according to user demand.
- **Practical simulation:** Practical simulations were conducted for scenarios using Python. A simulation environment similar to a real monitoring system was built to capture the performance variations under various network scenarios.

The rest of this paper is organized as follows. At first, the motivation and literature reviews are explained. Then, our main contributions with three new algorithms are described. The performance evaluation and analysis are presented. Finally, conclusions and future work are given.

Table 1: A table of acronyms

Abbreviations	Definition
CDN	Content delivery networks
MEC	Multi-access edge computing
RL	Reinforcement learning
ML	Machine learning
Cloud-based CDN	CCDN

(Continued)

Table 1 (continued)

Abbreviations	Definition
Vehicular ad hoc networks	VANET
Flying ad hoc networks	FANET
Content delivery on vehicular clouds	CDVC
Ad hoc on-demand distance vector	AODV

2 Methodology for Content Caching in Drone-Aided Ad Hoc Networks

In this section, we present a methodology for content caching in drone-aided ad hoc networks. First, the fundamental system model and procedures employed in our approach are described with data structures. Second, details for content caching are presented in pseudo code.

2.1 System Model

The construction of CDNs for drone-aided ad hoc networks entails the following phases: 1) ad hoc network configuration, 2) content placement, and 3) content replacement. Initially, there are N drones. Initially, any drone, D_i, is located at the center of the ith grid zone and has available capacity D_i^C set to zero. A drone serves and controls all communications from users in a particular region. This implies that the user can access content through the serving drone. Drone-aided ad hoc networks consist of N drones employing one of the proactive or reactive routing protocols. A specific node called a coordinator maintains the drone and content information in a table. S_X denotes the size of content X.

Fig. 1 shows the system model with the physical architecture for logical drone ad hoc networks. As shown in Fig. 1, a request for content is forwarded to the coordinator through drones. The coordinator searches the table and replies with drone ID (Identification). If the coordinator fails to search content requests in a table, it decides the serving drone with the proposed schemes in the following algorithms. A serving drone requests content to the original CDN server. Thus, the content is then delivered to the user through serving drones from either drone ad hoc networks or the original CDN server. A coordinator is accessible from all the drones and collects content information periodically or whenever a new content update occurs. In this section, we propose three algorithms for content placement in the coordinator node. A list of symbols for system and algorithm is given in Table 2.

The total delay of content delivery in the proposed scheme consists of delay from the original CDN server and one from drone-aided ad hoc networks. For the first content request, content should be delivered and hosted on the specific drone. And then, it is served to a host. So, the total delivery delay is the sum of transmission time from the original CDN server, drone decision time at the coordinator node and transmission time between the drone and user. But, as for the second request for the same hosted content, the delay of content delivery is limited to the same as the delay in transmission time between drone and users. Thus, the total delay is proportional to the miss rate for all requests that are not hosted in drone-aided ad hoc networks. On the other hand, total content delivery delay without the proposed scheme is accumulated by adding each transmission delay from the original CDN server to a user, repeatedly. Thus, total delay is proportional to the number of requests even though a user requests the same content.

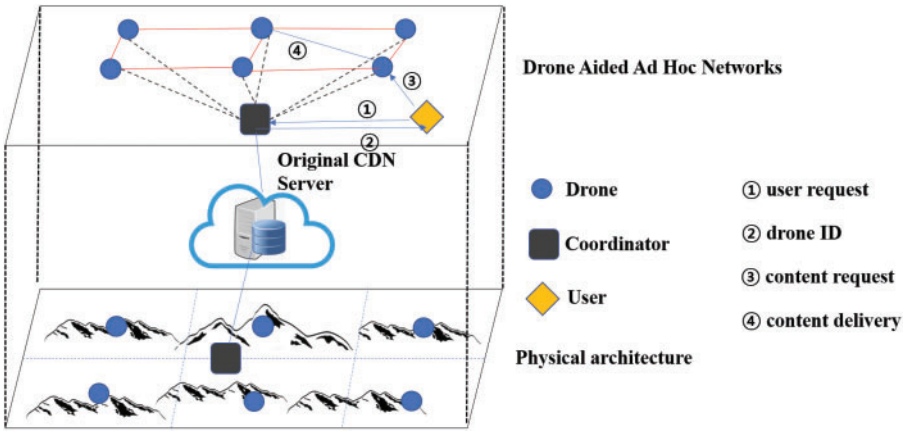

Figure 1: Illustration of network architecture

Table 2: A table of symbols

Symbols	Brief description
X and X'	An original content X and its partial segment
CT	Content management table in coordinator node
D_i	A drone with ID i
D_i^C	A capacity of D_i
S_X	Size of content X
$Tran_Delay(i,j)$	Transmission delay between drone i and j
$X.Timer$	Expiration timer for content X in a drone
FH_Timer	User defined expiration timer for a content

2.2 Algorithm 1: Greedy Algorithm

As a simple approach without computational complexity, the greedy algorithm is first applied for the basic procedures in Algorithm 1. When a coordinator node receives a request for content X, it searches the content management table and returns the corresponding drone ID. If no content exists for X, then the coordinator replies with $NONE$. As shown in Lines (3)–(4), if one of the drones has content X, content delivery is requested from D_{NID}. Otherwise, the content is transferred from the original server and recorded in one of the drones.

Algorithm 1: Greedy algorithm for content X

Initialization:

X: Content ID
NID: Drone ID
CT: Content management table
k: Drone serving a requesting user

(Continued)

Algorithm 1 (continued)

Algorithm:

1: **upon** Receiving request for X
2: $NID \leftarrow$ ***Search_Table(X)***
3: **if** NID exists in $Node\_List$ **then**
4: Request X to D_{NID}
5: **else**
6: $NID \leftarrow k$
7: **if** $D^C_{NID} >= S_X$ **then**
8: Record X in D_{NID}
9: $D^C_{NID} \leftarrow D^C_{NID} - S_X$
10: Update CT with NID and X
11: **else**
12: Call ***Find_Available_Node (X, NID)***
13: **end if**
14: **end if**

The procedure that determines the drone to host content X is described in Lines (7)–(12). The first candidate drone is a serving drone that minimizes content transmission time. However, this depends on the amount of storage available. If the remaining storage on the serving drone is larger than S_X, X is maintained in the serving node; otherwise, the $Find\_Available\_Node(X, NID)$ function is executed. This function searches for drones with an available storage larger than S_x. The search is performed starting with neighboring drones to distant ones considering transmission time. When no neighboring drones are found, the search procedure continues for drones within a two-hop neighborhood. If an available drone is found, X is maintained in that drone.

2.3 Algorithm 2: Timer and Segmented Content

Although Algorithm 1 based on the greedy approach provides basic procedures for CDN drone-aided ad hoc networks, there are some issues that need to be addressed. First, content replacement in Algorithm 1 is accomplished only when new content is requested. Second, prolonged content delivery is inevitable for far nodes. Third, because of the limited storage capacity of a drone, large content cannot be recorded near the user if other content is already recorded. To address these issues, we allowed multiple contents in networks, to cover the case that the requested content is located at a distance. However, because multiple content items are not desirable for drones, each content item is managed by a dynamic timer set to a value proportional to the request frequency. In addition, we introduced the concept of segmented content by recording partial content over multiple drones. The details of the enhanced algorithm are as follows:

In comparing Algorithms 1 and 2, some parts require further explanation. The first part consists of Lines (5)–(15), which allow duplicate recordings on multiple drones for identical content. In Line (5), the transmission delay for content X is measured. If this delay is longer than the threshold value, it is sufficient to allow for multiple records of content X. However, the limited storage requires checking the available capacity in Line (6). If a drone has sufficient available storage, it is recorded as a serving drone. Otherwise, partial content is recorded in the drone, as shown in Lines (10)–(14).

However, duplicate content is critical to drones with limited storage; therefore, each piece of content is dynamically managed by the timer. Unlike Algorithm 1 without a timer, each entry is

controlled by the timer such that if the timer expires, the content is automatically deleted from the drone. The timer value is determined by the frequency and source of requests. A higher value is assigned to nodes with more frequent requests from nearby users.

$$FH\_Timer = Default \times \left(\alpha \times Source\_Rate \times \frac{1}{Avg\_FS} + (1-\alpha) \times (1 - Source\_Rate) \times \frac{1}{Avg\_FO} \right) \quad (1)$$

As expressed in Eq. (1), the timer value is computed by considering the number of requests received from the serving users out of the total number of requests, where α is a weighting factor. This is denoted by *Source_Rate*. In addition, the average frequency of requests and time between requests, denoted by *Avg_FS* and *Avg_FO*, respectively, are considered by the timer. The former indicates how frequently users request content, whereas the latter indicates the interval within which to expect other requests. A weight factor α is used to distinguish requests from either its covering users or users covered by other drones. Through this factor, we can determine how much weight is given to the requests from its covering users which leads to reduce transmission delay.

Another outstanding feature is shown in Lines (16)–(24): When the available storage of drones is insufficient to record content, our algorithm finds the next node searching from neighboring nodes to distant nodes, which causes a long delivery time, particularly when networks become congested. This problem can be overcome by recording the content into available segments to shorten the delivery time. The number of segments is determined based on the storage availability on the drone. This is represented by Lines (17)–(32). After recording the feasible segments, neighboring nodes that can accommodate the remaining segments are required. This procedure continues until all the segments are recorded. If there is no available storage, the search procedure for the entire content begins, as shown in Line (28). This procedure is the same; however, the parameters for the node searching are different.

Algorithm 2: Timer and segmented content X

Initialization:

X: Content ID
NID: Drone ID
CT: Content management table.
k: Drone ID to serve the requesting user
Timer: Expiration time for the content
X': Partial portion of X

Algorithm:

1: **upon** Receiving request for X
2: $NID \leftarrow$ ***Search_Table (X)***
3: **if** *NID* exists in *Node_List* **then**
4: Request X to D_{NID}
5: **if** $Tran\_Delay(NID, k) >= Threshold$ **then**
6: **if** $D_k^C >= S_X$ **then**
7: Record X in D_k
8: $X.Timer \leftarrow FH\_Timer$
9: Update *CT* with k and X

(Continued)

Algorithm 2 (continued)

```
10:       else
11:          Record X' in D_k
12:          S_X' ← S_X − D_k^C
13:          Update CT with k and X'
14:          Call Find_Available_Node (X − X', k)
15:       end if
16:    end if
17: else
18:    NID ← k
19:    if D_NID^C >= S_X then
20:       Record X in D_NID
21:       D_NID^C ← D_NID^C − S_X
22:       Update CT with NID and X
23:    else
24:      if D_NID^C != 0 then
25:         S_X' ← S_X − D_NID^C
26:         Record X' in D_NID
27:         Update CT with NID and X'
28:         Call Find_Available_Node (X − X', NID)
29:      else
30:         Call Find_Available_Node (X, NID)
31:      end if
32:    end if
33: end if
34:
35: Periodically Call Content_Relocation (X)
```

Fig. 2 illustrates the different features of the content management table. First, a timer is added to the case of Algorithm 2. Moreover, in the case of content ID "1", one entry exists instead of the three entries in Algorithm 2 for segmentation. Furthermore, for partial segments, the function for the search operation *Search_Table*(X) is different from that in Algorithm 1. Multiple drone IDs must be returned as content. Additionally, if multiple contents exist, a drone ID with a longer timer value is selected, and the corresponding ID is returned for content delivery.

Content ID	Drone ID
1	3
2	5
3	7

(a) Example of CT in an Algorithm 1

Content ID	Drone ID	Timer
1.1	3	2
2	5	4
1.2	2	4
1.3	7	3

(b) Example of CT in an Algorithm 2

Figure 2: Example of CT

The last part in Algorithm 2 is for function (*Content_Relocation*(X)) in Line (35). This is periodically called to decide which node is the best one to reduce content delivery time. Based on this decision, content is relocated to the corresponding node. The first step is to measure content

delivery time for content X for current user requests. Thus, content delivery time from drone 0 to drone n is measured. However, since the content is now segmented, this function is to relocate the largest segmented block rather than the whole content. This can be represented where content X is assumed to be drone 0. In addition, this computation will continue until drone ID reaches NUM_D.

After computing the $Delay[i]$ with above code in Fig. 3, the best candidate drone e is chosen by Eq. (2) for content X by search total delay from all drones. In addition, if D_e^C is not larger than S_X, content X is moved to drone c. Otherwise, content X is kept on the current drone.

$$e \leftarrow \arg\min_i (Delay[i]) \tag{2}$$

```
for (i = 0; i < NUM_D; i++)
    for ( j = 0; j < NUM_D; j++)
        if (Find(X,j) == TRUE)
            Delay[i] = Delay[i] + Tran_Delay(i,j)
```

Figure 3: Code for measure delay

2.4 Algorithm 3: Drone Movement

A previously segmented record is a good approach for addressing the limited storage capacity of a drone. However, this also results in a long delivery time because some segments can be recorded on distant nodes. To reduce delivery time and locate content near a user, the most efficient scheme is to move drones to a nearby area, utilizing the mobility of the drone. However, the movement of the drones leads to holes or voids; therefore, additional drones must be deployed. In this study, two drones were deployed per grid area, designating one drone as the primary and the other as the secondary. If the primary drone moves to other areas, the secondary drone covers the corresponding area. Algorithm 3 outlines the detailed procedures used to determine the drone movement.

Algorithm 3: Drone movement for content X

Initialization:

$DC_X \leftarrow$ Delivery time for content X
$NID \leftarrow$ Drone ID with new content X
$NUM\_D \leftarrow$ Total number of drones
$Temp \leftarrow 0$
$TempID \leftarrow$ NID

upon Updating CT with X at drone NID
  call Drone_Movement (X, NID)

Function: Drone_Movement (X, NID)
for $i \leftarrow 0, i < NUM\_D$ **do**
  if X is in D_i **then**
    $DC_X \leftarrow DC_X + Tran\_Delay(NID, i)$
    if $Tran\_Delay(NID, i) >= Temp$ **then**
      $Temp \leftarrow Tran\_Delay(NID, i)$
      $TempID \leftarrow i$
    end if
  end if

(Continued)

Algorithm 3 (continued)

12: **end for**
13: **if** $DC_X >= Threshold$ **then**
14: Move D_{TempID} to near D_{NID}
15: Secondary node take over D_{TempID}
16: **end if**

Whenever content X is newly updated in a CT on the drone with NID, the demand for drone movement is checked by calling a function that determines whether any drone containing X should move to a nearby drone with NID. Instead of the simultaneous movements of multiple drones, using only one drone does not incur computational complexity. To accomplish this, whenever content is recorded in CT, the transmission delay from the recent $D_N ID$ to any drone with content X is measured, as shown in Lines (4)–(6), thereby obtaining the total cost of content transmission for content X. From Lines (7)–(9), the farthest node with content X is determined by comparing the transmission delay with the existing value. As a result, the total cost and drone ID with the longest transmission are computed and identified.

If the sum of the transmission delays for content X is greater than the predetermined threshold value, as indicated in Line (13), the farthest node is moved toward the drone with NID to reduce the transmission delay. A coordinate node requests a movement from the corresponding node by sending feasible positional information. Despite drone movement, if the transmission delay still exceeds the predetermined value, another drone moves toward the drone with NID. This procedure continues until the transmission delay is lower than the threshold.

3 Performance Evaluation

In this section, we present the comparative performance evaluation results obtained through simulations. We employed the network simulator NS3 (Network Simulator 3) to build drone-aided ad hoc networks and control the traffic model. Most of the parameters for the simulation followed the acceptable scenarios where the default value is determined by the ad hoc on-demand distance vector (AODV) protocol. For the comparison metric, the content delivery time and drone capacity utilization were measured under several scenarios, including user request frequency and content size. For the evaluation, we selected a user and content randomly. It means any specific user was selected for one of the contents. Without regard to this request, other randomly selected users can request other content. So, multiple requests by users for different contents are assumed in the simulation. We ran the simulation for 3600 s. The simulation was conducted 100 times considering a 95% confidence interval. Because there is no existing scheme for CDN in drone-aided ad networks, performance evaluation was conducted using the three algorithms presented in this study. Details of simulation paramters are given in Table 3.

Table 3: Simulation parameters

Parameter	Value
Routing protocol in drone networks	AODV
Transmission range of drone	200 m
MAC protocol	IEEE 802.11b

(Continued)

Table 3 (continued)

Parameter	Value
Number of grid zones	10
Drone storage capacity	32 G
Weight value (α) in Algorithm 1	0.7
Default in Algorithm 1	2 s

3.1 User Request Interval

The first simulation scenario evaluated performance as a function of user request frequency with a fixed drone storage capacity and content size varying between 1 and 4 GBytes. We assumed that a randomly selected user could request content of various sizes from a serving drone. The average delivery delay and capacity utilization were then measured. Figs. 4 and 5 present the comparative evaluation results. In both figures, Alg1 represents Algorithm 1 and Alg2 and Alg3 represent Algorithms 2 and 3, respectively.

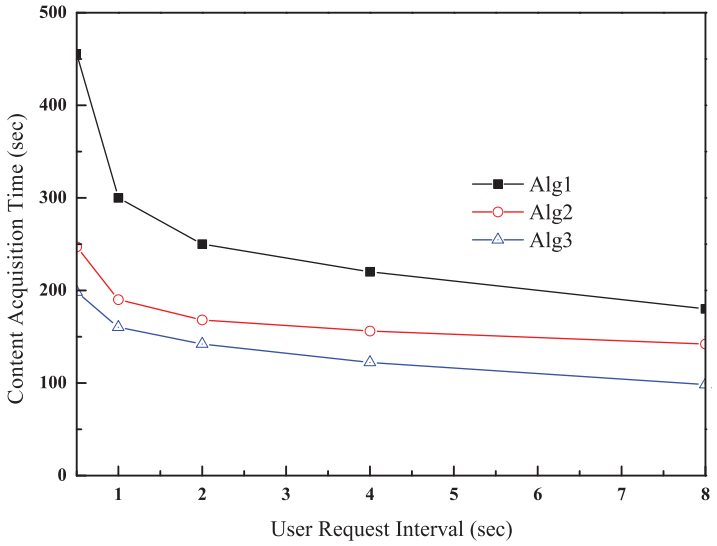

Figure 4: Content acquisition time as a function of content request interval

As illustrated in Fig. 4, Algorithm 1 has the longest delivery time for content. Usually, because the user request frequency is low, a longer content acquisition time is required. The long delivery time is caused by frequent caching misses. In addition, because AODV takes the shortest path for delivery, frequent content requests make this path congested. Algorithms 2 and 3 exhibit shorter content acquisition times than Algorithm 1. Multiple contents reduce caching misses near the drone. Consequently, a low caching miss leads to a short delivery time by providing content that is as close to the user as possible. Because multiple content placement is dynamically determined by the content delivery time, this is the main factor that reduces the acquisition time. However, there is the issue of capacity utilization in Algorithms 2 and 3 because multiple contents may be maintained on the drones. This implies that the content can be placed at a far node when running out of storage. Despite this

issue, Algorithm 3 has the best content acquisition time because of drone mobility. As the number of user requests increases, more drones move to areas near the user, thereby providing a huge storage capacity for frequent content requests, which drastically reduces content acquisition time.

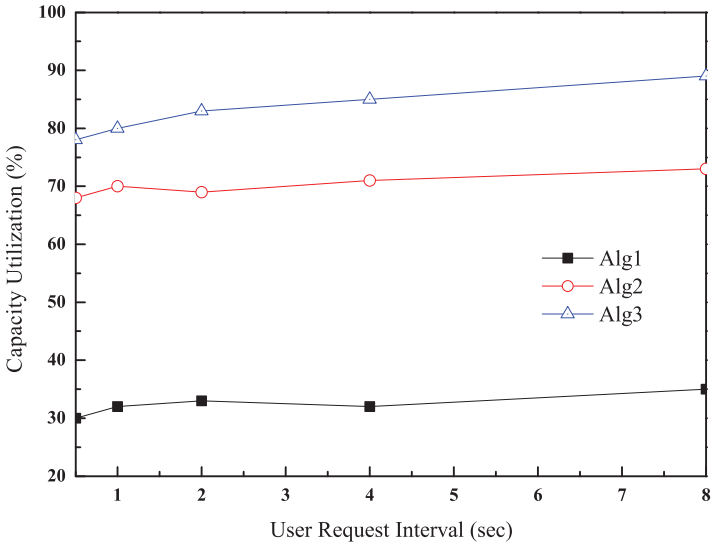

Figure 5: Capacity utilization as a function of content request interval

As compared to Algorithm 1, there are outstanding features in Algorithm 2. The first is for a dynamic timer for content that is controlled by how many and how frequently the user requests. So, the user request interval becomes longer and longer, it is more likely to accommodate new content. Similarly, multiple content with longer timer can be located near the user, content acquisition time is reduced as shown in Fig. 4. The content relocation triggered by function *Content_Relocation*(X) is affected by the current available storage. If the storage is occupied by other contents, this function does not contribute to reducing content acquisition time. In other words, if the user request interval is short, content relocation is rarely conducted. On the other hand, drone movement is not impacted by the available storage. As a result, it is noticeable that outstanding features in Algorithm 2 do not ensure short content acquisition time due to limited storage on a drone. Consequently, since a dynamic timer leads to vacant storage on a drone, it is a very crucial part of Algorithm 2.

The capacity utilization in Fig. 5 indicates the amount of storage required by content near a user. This is computed by dividing the size of the entire content by the size of the requested content on the drone. Therefore, a higher value indicates that more content is placed on the serving drone for a user. Compared with Algorithm 1, Algorithms 2 and 3 show higher capacity utilization because the concept of segmented content is introduced. In Algorithm 1, if the available capacity cannot accommodate new content, it remains unoccupied until smaller content that can be accommodated by the remaining size is requested. Based on this operation, the capacity utilization in Algorithm 1 does not match those of Algorithms 2 and 3, which address this issue by using segmented content on multiple nodes. Segmented content leads to higher capacity utilization by accommodating partial content, even though the remaining storage is not greater than the content size. Specifically, drone movement contributes to an increase in capacity utilization by forming content clusters. The more drones are moved to a specific area, the higher the capacity utilization achieved. Generally, capacity utilization increases as the number of user requests increases, except in Algorithm 1.

3.2 Content Size

In this evaluation, we measured performance for a fixed user with varying content sizes requested at intervals of 1 s. For the simulations, the content size was set to a randomly chosen value between one and the maximum size indicated by the x-axis value. For example, 8 GBytes along the x-axis in 6 indicates that the content size ranges from 1 to 8 GBytes. Fig. 6 shows the content acquisition time for content size. As the content size increases, the content acquisition time also increases for all algorithms. Content acquisition time is affected the most in Algorithm 1 because a large amount of content can be placed at a distant node with large available storage. Based on these features, a longer acquisition time is obtained as content size increases. The impact of large-sized content is reduced in Algorithms 2 and 3 by placing partially segmented content on the drone. However, multiple contents on drones also cause a long acquisition time, reducing the differences among the algorithms. Among the proposed schemes, Algorithm 3 has the shortest acquisition time because of drone movements. Consequently, several drones around a user provide a large content capacity, thereby increasing the differences between algorithms in the case of large content measurements.

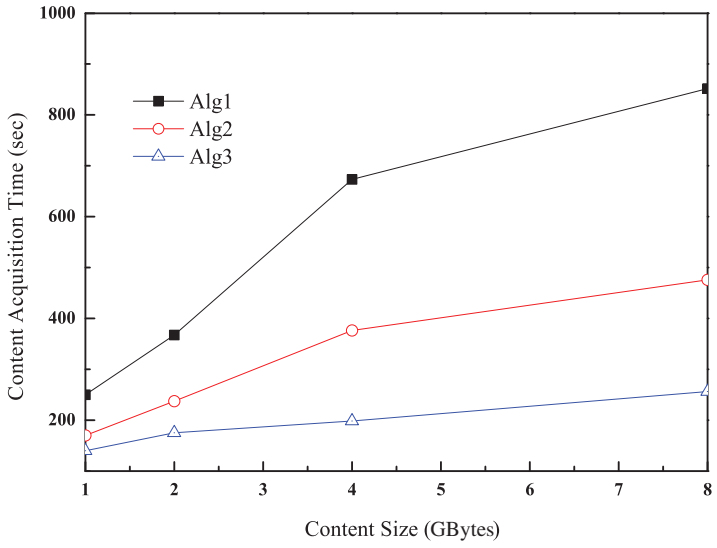

Figure 6: Content acquisition time as a function of content size

Similar to the analysis in Fig. 6, Fig. 7 shows the content utilization of the three algorithms. As mentioned above, the value in Algorithm 3 is the highest and is least affected by content size. In contrast, the lowest value is observed in Algorithm 1 with content size of 8 GBytes. In this case, content is rarely placed near a user because other content is likely to be placed according to the order of user request. However, the accuracy of Algorithm 2 is lower than that of Algorithm 3 and higher than that of Algorithm 1. First, multiple contents lead to higher values than those in Algorithm 1 in the networks. However, this content can be removed if the timer expires. Therefore, a new user request does not arrive before the timer expires, and the content must be delivered from the other node. This problem is addressed by the drone movement in Algorithm 3.

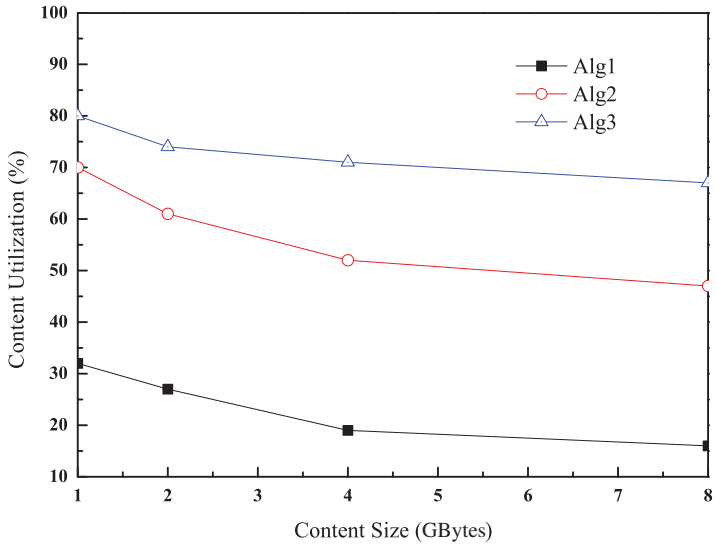

Figure 7: Capacity utilization as a function of content size

3.3 Drone Storage Capacity

Unlike previous simulation scenarios on content size, drone storage capacity also has an impact. For this simulation, the content size was set between 1 and 8 GBytes, and the user frequency interval to 1 s. The drone storage capacity ranged from 16 to 64 GBytes. As shown in Fig. 8, all algorithms exhibit the shortest content acquisition time when a large storage capacity is assigned to the drone. A large difference is observed in Algorithm 1 for a small drone storage capacity. However, the difference between the algorithms decreases for a large capacity, as most of the content is placed near a serving drone, as shown in Fig. 9. However, the general patterns for content acquisition time and capacity utilization are the same as those in the previous case.

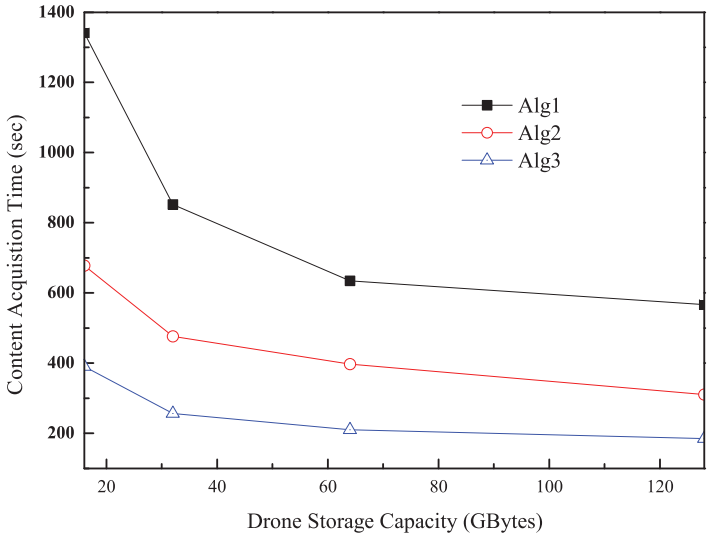

Figure 8: Content acquisition time as a function of drone storage capacity

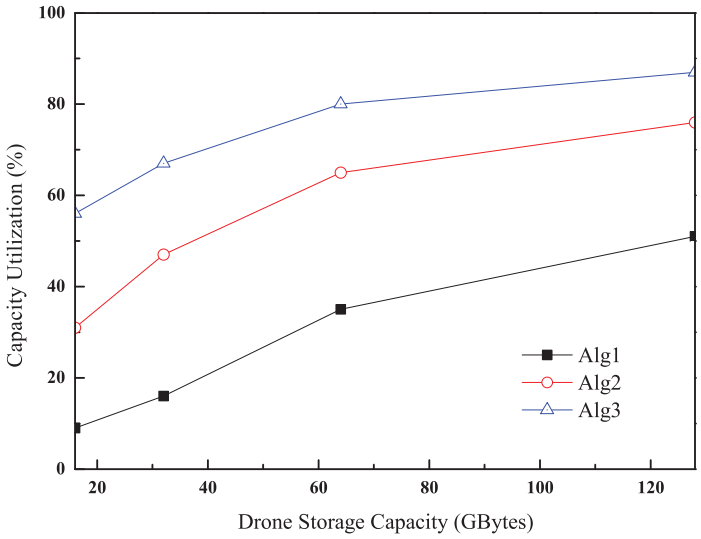

Figure 9: Capacity utilization as a function of drone storage capacity

Furthermore, Algorithm 3 does not differ by much. This is because drone movement has the same impact as a large drone capacity. As more drones move, the content acquisition time decreases and capacity utilization increases. This effect is critical in the case of small drone storage capacity. Capacity utilization is improved as the drone storage capacity increases because of the partially segmented content in both Algorithms 2 and 3. Furthermore, the improvement in capacity utilization in Algorithm 1 is drastic as the requested content is placed on the serving drone as much as possible. This implies that Algorithm 1 is feasible when a drone has sufficient storage capacity without complicated operations.

4 Conclusions

In this paper, we presented three new content caching algorithms for drone-aided ad hoc networks where infrastructure is not available for CDN. These three algorithms were designed and developed, primarily considering the constraints of the drone and its mobility. Simulation results revealed that the proposed schemes can be implemented in drone-aided ad hoc networks.

In future work, we will design and conduct several simulation cases with practical scenarios, such as military operations. In addition, the current algorithm will be extended to include the next content prediction using machine learning algorithms. In addition, there are some issues with the optimization to minimize the acquisition time for several functions such as *Content_Relocation*(X) because the current algorithm is designed by respective content not the whole one.

Acknowledgement: We would like to express our sincere gratitude to the reviewers for their comments. Their insights and suggestions are very valuable.

Funding Statement: This result was supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) (2021RIS-004) and the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. RS-2022-II221200, Convergence Security Core Talent Training Business (Chungnam National University)).

Author Contributions: Data collection: Yong Beom Park, Jian Kim; analysis and interpretation of results: BeomKyu Suh, Ismatov Akobir; draft manuscript preparation: Ki-Il Kim. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] J. Sahoo, M. A. Salahuddin, R. Glitho, H. Elbiaze, and W. Ajib, "A survey on replica server placement algorithms for content delivery networks," *IEEE Commun. Surv. Tutor.*, vol. 19, no. 2, pp. 1002–1026, 2017. doi: 10.1109/COMST.2016.2626384.

[2] Q. Jia, R. Xie, T. Huang, J. Liu, and Y. Liu, "The collaboration for content delivery and network infrastructures: A survey," *IEEE Access*, vol. 5, pp. 18088–18106, 2017. doi: 10.1109/ACCESS.2017.2715824.

[3] M. A. Salahuddin, J. Sahoo, R. Glitho, H. Elbiaze, and W. Ajib, "A survey on content placement algorithms for cloud-based content delivery networks," *IEEE Access*, vol. 6, pp. 91–118, 2017.

[4] Y. Gao *et al.*, "Video content placement at the network edge: Centralized and distributed algorithms," *IEEE Trans. Mob. Comput.*, vol. 22, no. 11, pp. 6843–6859, 2023.

[5] H. Li *et al.*, "Intelligent content caching and user association in mobile edge computing networks for smart cities," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 1, pp. 994–1007, 2024. doi: 10.1109/TNSE.2023.3312369.

[6] P. Lin, Z. Ning, Z. Zhang, Y. Liu, F. R. Yu and V. C. M. Leung, "Joint optimization of preference-aware caching and content migration in cost-efficient mobile edge networks," *IEEE Trans. Wirel. Commun.*, vol. 23, no. 5, pp. 4918–4931, 2024. doi: 10.1109/TWC.2023.3323464.

[7] N. Nomikos, S. Zoupanos, T. Charalamous, and I. Krikidis, "A survey on reinforcement learning-aided caching in heterogeneous mobile edge networks," *IEEE Access*, vol. 10, no. 4, pp. 4380–4413, 2022. doi: 10.1109/ACCESS.2022.3140719.

[8] Z. Lu *et al.*, "A survey on reinforcement learning-aided caching in heterogeneous mobile edge networks," *IEEE Netw.*, vol. 34, no. 6, pp. 4380–4413, 2020.

[9] A. Sadeghi, G. Wang, and G. B. Giannakis, "Deep reinforcement learning for adaptive caching in hierarchical content delivery networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1024–1033, 2019. doi: 10.1109/TCCN.2019.2936193.

[10] V. Kirilin, A. Sundarrajan, S. Gorinsky, and R. K. Sitaraman, "RL-Cache: Learning-based cache admission for content delivery," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2372–2385, 2020. doi: 10.1109/JSAC.2020.3000415.

[11] A. Masood, D. Q. Tuan, D. S. Lakew, N. Dao, and S. Cho, "A review on AI-enabled content caching in vehicular edge caching and networks," in *Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, IEEE, 2023.

[12] X. Wang and Y. Li, "Cache space efficient caching scheme for content-centric mobile ad hoc networks," *IEEE Syst. J.*, vol. 13, no. 1, pp. 530–541, 2019.

[13] X. Wang and Y. Li, "Content delivery based on vehicular cloud," *IEEE Trans. Vehicular Technol.*, vol. 69, no. 12, pp. 2105–2113, 2020. doi: 10.1109/TVT.2019.2959799.

[14] T. Duong, Z. K. K. Kim, M. Bui, and N. Vo, "UAV caching in 6G networks: A survey on models, techniques, and applications," *Phys. Commun.*, vol. 51, no. 1, 2022, Art. no. 101532. doi: 10.1016/j.phycom.2021.101532.

[15] T. Bilen and B. Canberk, "Content delivery from the sky: Drone-aided load balancing for mobile-CDN," *EAI Endorsed Trans. Ind. Netw. Intell. Syst.*, vol. 9, no. 30, pp. 1–7, 2022.