


Please cite the Published Version

Naeem, Muhammad Ali, Bashir, Ali Kashif  and Meng, Yahui (2025) Dynamic cluster-based cooperative cache management at the network edges in NDN-based Internet of Things. Alexandria Engineering Journal, 125. pp. 297-310. ISSN 1110-0168

DOI: <https://doi.org/10.1016/j.aej.2025.03.131>

Publisher: Elsevier

Version: Published Version

Downloaded from: <https://e-space.mmu.ac.uk/639593/>

Usage rights:  [Creative Commons: Attribution-Noncommercial-No Derivative Works 4.0](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Additional Information: This is an open access article which appeared in Alexandria Engineering Journal, published by Elsevier

Enquiries:

If you have questions about this document, contact openresearch@mmu.ac.uk. Please include the URL of the record in e-space. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from <https://www.mmu.ac.uk/library/using-the-library/policies-and-guidelines>)



Dynamic cluster-based cooperative cache management at the network edges in NDN-based Internet of Things

Muhammad Ali Naeem^a, Ali Kashif Bashir^b, Yahui Meng^{a,*}

^a School of Science, Guangdong University of Petrochemical Technology, Maoming 525000, China

^b Department of Computing and Mathematics, at Manchester Metropolitan University, Manchester, United Kingdom

ARTICLE INFO

Keywords:

Caching
Internet of Things
Information-centric networking
Named data networking

ABSTRACT

The Named Data Networking (NDN) has been postulated as a new paradigm for the Internet of Things (IoT). It utilizes its data-centric networking model where content is acquired by name instead of position. Meanwhile, in NDN networks, caching is included in the nodes for temporary storing or user request responses. However, nodes in IoT networks have small cache storage compared to the huge amount of content transmitted, which poses challenges in efficiency. This work introduces a new caching policy for NDN-IoT systems called Dynamic Clustering-based Cooperative Caching (DCCC). DCCC determines and stores the content hitting rates concerning the user desires, dynamic threshold, and dissemination of cooperation with inter-node adaptive clusters. In detail, the results from growing numbers of simulations show that DCCC achieves better outcomes than prior caching schemes in average cache hits per node, hops reduction per node, and the delay of content delivery.

1. Introduction

Currently, the global advancement of Internet of Things (IoT) applications, along with an even greater boost in Internet usage, has generally catalyzed the greatest level of end-user demand [1,2]. IoT is discussed globally as a revolutionary technology that offers effective connections and spreads accurate real-time information back and forth across the world [3,4]. To support these applications, a fair number of architectures have been proposed, among which Named Data Networking (NDN) is promising as a candidate to become the architecture of the future Internet, especially for IoT-based systems [5].

As useful as IoT devices may be, they are also very resource-limited concerning fields like computation, power, or memory [6]. These limitations question the applicability of NDN's default caching policy to cache anything anywhere, where all transmitted content is cached at every node in the data delivery path [7]. As this strategy provides very high data availability, it is infeasible for IoT settings due to resource scarcity and, most importantly, the storage capabilities of IoT nodes [8, 9]. Further optimizing NDN caching strategies to be more acceptable and within capacity on the IoT front is necessary [10,11].

The first and possibly the most significant problem is identifying where different content in the network should be cached for the best overall system performance. In addition, it is important to determine

how long content should be stored at a network node to optimize benefits and limit resource consumption [12,13]. Although so many strategies have been provided to deal with these problems [5,14–19], most of these strategies do not meet the special conditions and constraints of IoT-based environments. For example, caching unnecessary and repeated information brings extra expensive traffic in the network, memory consumption, and a time delay in data access that works against the purpose of caching [20].

As a result, determining the best caching solution becomes critical. NDN-based caching should handle existing caching issues while also successfully addressing restrictions in situations such as IoT-based applications [21,22]. Caching redundant content, on the other hand, may increase network load and data retrieval latencies. As a result, the best caching solution can minimize overall network latency by enhancing demand for data availability [16,23]. In this situation, the best caching solution may identify the most frequently downloaded items and cache them for a long period at suitable network nodes, so subscribers can easily and quickly get them in the future during their sessions [24,25]. As a result, the distance between subscribers and data provider nodes is reduced, and data fetching time is cut dramatically. So, the gap for improving the entire caching network remains, and to cover it, we designed a Dynamic Clustering-based Cooperative Caching (DCCC) technique to improve overall caching performance. The key goals of this

* Corresponding author.

E-mail addresses: malinaeem@gdupt.edu.cn (M.A. Naeem), dr.alikashif.b@ieee.org (A.K. Bashir), mengyahui@ggdupt.edu.cn (Y. Meng).

<https://doi.org/10.1016/j.aej.2025.03.131>

Received 21 October 2024; Received in revised form 25 January 2025; Accepted 29 March 2025

Available online 16 April 2025

1110-0168/© 2025 Published by Elsevier B.V. on behalf of Faculty of Engineering, Alexandria University. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

research are to improve cache usage at network edges and reduce data delivery delays.

To reach these aims, we introduced a dynamic cluster-based caching model that solves a number of important problems in the study of how content is distributed and how well networks work. By tackling these challenges, our approach makes multiple significant contributions to the field. The following is the study's main contribution:

- **Popularity differentiation:** The capacity to discern popular content from less popular content is a significant difficulty in content distribution systems. Our solution uses dynamic criteria to precisely identify content based on its level of popularity, ensuring that network resources are directed to high-demand content in the most effective manner possible.
- **Interest-Based Clustering:** Our strategy's dynamic clustering method efficiently addresses the issue of grouping network nodes with similar Interests and preferences. Our technology dynamically produces clusters by taking into account Interest frequency and user preferences. Furthermore, the cluster head is chosen based on the node's relevance and the proximity of the subscribers, guaranteeing effective and localized content distribution.
- **Cache Status Sharing:** We take it a step further by suggesting a method for determining the most essential node inside a cluster as the cluster head. In addition to keeping its cache state, this cluster head cooperatively distributes it with surrounding nodes. This innovation improves data accessibility and cache hit rates.
- **Adaptive Caching:** A key component of our approach is the backup caching mechanism, which addresses the issue of efficient cache allocation. When the cluster head's cache is full, it effectively evicts and moves less popular content to the most important neighbor node while recognizing and caching high-popularity content. Adaptive caching constantly optimizes the network to serve the most popular information, increasing network effectiveness overall.

The rest of the article is organized as follows: Section II defines the related studies. Section III describes the proposed caching strategy. Section IV provides details about the evaluation of the proposed caching model. Section V provides the evaluation results. Finally, we concluded the paper in Section VI.

2. Related studies

The caching module is regarded as the most important component in NDN for improving the entire communication system since it is important for splitting up data from its physical locations and delivering quick data dissemination over the network [26,27]. From an IoT perspective, the communication process consumes more energy than IoT-based tasks performed on devices [28]. In this context, NDN-based caching plays an important role in lowering the frequency of IoT node accesses and minimizing total access time. As a result, the NDN-based caching module can be used in an IoT-based network to speed up the network by storing temporary data [12]. Several studies have recently been offered to effectively increase the performance of NDN-based IoT setups. Hahm et al. [29] investigated a sleep-based caching scheme for caching transmitted material in an NDN-based IoT environment. It tries to increase suitable data availability in order to improve network energy efficiency. To save energy, the contents are cached at a node while it is awake, while the node is mostly asleep the rest of the time. However, it employs two replacement strategies that increase processing complexity and data redundancy by caching contents across many nodes. Meddeb et al. [30] introduced a caching system that uses a freshness technique to determine the validity of cached contents while minimizing caching costs. However, it raises the complexity overhead. Furthermore, Amadeo et al. [31] developed an NDN-based caching technique to increase the performance of IoT-based networks in their study. Caching actions are performed using freshness and data popularity values as

characteristics. As a result, caching actions are carried out autonomously on the edge node. However, because of the autonomous caching activities conducted by each node, data replication is performed at numerous places.

Rahul et al. [32] proposed a smart caching technique that uses an extra table at each node to store information. This table stores all inbound requests for each content in order to determine popularity. Content is cached based on the popularity and freshness of the data. However, it promotes data redundancy and multiple duplications at many nodes. Miwa et al. [31] offer a cooperative caching technique in which a timer is coupled with the network to determine the lifespan of an item. If many nodes require the same content, it is cached at all of them via a cooperative updating technique. Caching contents at all nodes along the data routing channel promotes data redundancy. Furthermore, it updates information on the contents even if the contents have expired, consuming more energy. Similarly, Asmat et al. [33] presented a caching technique for NDN-based IoT called Central Control Caching (CCC). Autonomous systems are deployed, and a central node is chosen to cache incoming data. However, it includes numerous tables to execute a single caching decision, which increases complexity. Naeem et al. [34] offer an NDN-based caching approach called Efficient Popularity-aware Probabilistic Caching (EPPC) to improve the performance of IoT networks in a recent study. EPPC is broken into three stages: data selection, caching, and replacement. It chooses the most frequently requested contents and caches them at the leading node, which is associated with a greater number of neighbor nodes. Dinh et al. [35] proposed an energy-efficient way to cache data in which the energy used for caching and retrieving data is used as a reward to measure how energy-efficient the method is. The author suggests data insertion and replacement algorithms based on the caching incentive to improve the energy efficiency of caching decisions. However, it raises the complexity overhead.

Amadeo et al. [36] offer a Popularity-aware Closeness (PoC) caching approach in which contents are chosen based on the number of incoming subscriber requests. If content has the most retrieval, it is designated as popular and is advised to be cached at the nearest centrality node. Finding the most popular content, on the other hand, has a higher complexity overhead. Baltagiannis et al. [37] offer an energy-aware caching approach called Fault-tolerant Probabilistic Caching in a recent paper. When making caching decisions, it takes sensor defects and energy consumption into account. On the other hand, it raises the communication overhead required to detect sensor problems. On the other hand, clustering techniques offer significant advantages in caching NDN-based IoT environments, particularly in optimizing cached data access for network efficiency and efficient use of scarce resources [38]. When a node is grouped in clusters, caching decisions can best be made through collaboration within the cluster, eliminating repetition and allowing faster access to data. Clusters provide an opportunity to perform cooperative caching, which means the representatives of clusters control cache decisions, including what should be cached and where it should be cached based on user interests and the existing network conditions. This approach also reduces the amount of data transmitted over the networked space, has low latency as long-distance communication is not always necessary, and is economical as power is saved through less traffic across the network. In addition, clustering improves scalability, and thus NDN-based IoT networks will better cope with increasing IoT traffic load while providing high performance and resource utilization [39–42].

As a result, in order to optimize the cache management method, the proposed caching strategy takes into account both the popularity of the transmitted content and the caching site with the help of clustering. It caches the contents collaboratively to solve the drawbacks of previous caching systems. It offers novel solutions to long-standing problems in content distribution and network optimization research. Our proposed strategy provides novel approaches to long-standing issues in content distribution and network optimization research. We greatly enhanced

content availability, user experience, and network efficiency by implementing dynamic threshold, Interest-based clustering, cache status sharing, and adaptive caching. The addition of dynamic threshold, interest-based clustering, cache status sharing and adaptive caching has significantly improved content availability, user experience, and network efficiency. By putting our technique into practice, we can transform how content is distributed and controlled inside networks, raising the bar for quality in research. The next sections go over the suggested caching strategy in depth.

3. Dynamic cluster-based cooperative caching

This section provides comprehensive details of the proposed caching strategy. As mentioned in the above sections, several problems still exist in NDN caching that hinder its full benefits for IoT-based networks. Therefore, for NDN-based IoT networks to fully benefit from caching, they need to implement a better caching strategy that can make caching work better overall. Therefore, to efficiently manage cache at distributed locations, we proposed a Dynamic Clustering-based Cooperative Caching (DCCC) strategy that can reduce communication overhead and data downloading latency. It minimizes the data redundancy ratio and improves the utilization of cache storage. DCCC is a three-fold caching strategy that includes data collection, content placement, and data caching mechanisms.

3.1. Data collection mechanism

In literature, studies show that popular content has a significant impact on caching capabilities. Sometimes, popular content can consume extra cache storage and slow down the cache hit rate when cache storage is available. Similarly, the popularity of content is dynamic concerning subscribers' Interests. Consequently, a static dynamic threshold is not appropriate because it assigns equal importance to less popular and highly popular content. This can result in increased resource usage (cache) by using less popular content instead of more popular content. Therefore, in DCCC, the cluster head nodes will cache content that carries higher popularity values compared to the local average popularity of cached content when the cache space has overflowed. To enhance cache utilization and avoid redundant data caching, popular content will be cached only at selected nodes. Each node is associated with a table called the Dynamic Threshold-based Popularity Table (DTPT), which maintains dynamic threshold values. This table (Table 1) contains the name of each content, the number of contents, the number of Interests for each content item, and the total number of Interests generated for all content.

Currently, a huge number of contents are transmitted on the network, but only a small number of contents are frequently fetched by a large number of subscribers. Therefore, popular content has a significant impact on improving overall caching performance by reducing data retrieval latencies. Thus, designing an efficient Data Collection Mechanism (DCM) by deploying an effective data collection algorithm is of

utmost importance. To measure the dynamic threshold value, each content is weighted with a specific value equivalent to the subscribers' Interest count, indicating how many times content item is retrieved within a given cycle. Consequently, the threshold value is calculated using the Weighted Average (WA), which can be determined using the following equation. Let $G(W, X)$ is a network graph where all the nodes represent the total number of contents X with their total weight W and it can be denoted by $W(X)$ where $X = \{x_1, x_2, \dots, x_n\}$ is the set of content for them the subscribers' Interests are generated to fetch them from the network and $W = \{w_1, w_2, \dots, w_n\}$ is the set of weights assigned to the content according to the subscribers' Interests. The Interests frequency can be denoted by $I(X)$ where $I = \{i_1, i_2, \dots, i_n\}$ that shows the number of contents downloaded at a network in a given cycle. Therefore, the weight at each node represents the Interests frequency or we can say it is equal to the Interests frequency and it can be denoted by $W(X) = I(X)$. Thus, the Dynamic Threshold (DT) can be measured by taking the accumulation of all weights and it can be calculated through the following equation:

$$DT = \frac{(w(x_1) + w(x_2) + I + w(x_n))}{X}(\alpha) \tag{1}$$

Eq. 1 can be simplified and DT can be calculated by following Eq. 2.

$$DT = \frac{\sum_1^n w(x_i)}{\sum_1^n x_i}(\alpha) \tag{2}$$

where α shows the given cycle in which the threshold is calculated. For the selection of popular content, the WA is calculated between the total number of contents that were recently fetched by the subscribers in a given cycle and the total number of Interests received for all contents. The outcome value is considered the dynamic threshold. In Algorithm 1, the dynamic threshold is determined, where TW shows the total weight that can be defined as:

$$TW = \{w(x_1) + w(x_2) + I + w(x_n)\} \tag{3}$$

Therefore, according to Algorithm 1, the DT is calculated at a node v_i to differentiate between the higher popular content and the less popular content. Based on the threshold value derived from Algorithm 1, content is considered popular or non-popular. If content shows its Interest frequency is higher than the threshold value the content will be chosen as a popular one and the selected names of content are pasted in another table called Popular Data Table (PDT). As the content passes through the node, the names are compared with names stored at the PDT table if the popular content x_i matches with its name at specific node v_i the content x_i is recommended to be cached at the node v_i . Let's assume that two subscribers' sub-1 and sub-2 sent multiple Interests such as the total weight is equal to 40 Interests to download content x_1 and content x_2 from a network node v_i . As we know the $w(x_1) = I(x_1)$ and when we separate the weight for each content, the weight of x_1 is calculated as $w(x_1) = 27$ while the weight of content x_2 is measured as $w(x_2) = 13$.

Table 1
Popularity Table.

Popular Data Table				
C-Name	I-Count	Threshold	Popularity	S-Content
x_1	27	20	27	Popular
x_2	13	20	13	Non-popular

Algorithm 1. Calculate Dynamic Threshold

Input: Total weights or total number of received Interest
Output: Dynamic Threshold Value
Procedure: Dynamic Threshold (v)

- 1: $TW = 0$
//TW shows the total weight
- 2: $Count = 0$
3. **for** x in v **do**
4. $I(x) = I_{count}(x)$ // Interest count for content (x)
5. $w(x) = I(x)$
6. $TW += 1$
7. **if** $count == 0$ **then**
8. **return** node is empty
9. $DT = TW/count$
10. **return** DT

Thus, using Eq. 1, the threshold value can be calculated as:

$$DT = \frac{w(x_1) + w(x_2)}{x_1 + x_2} = \frac{27(1) + 13(1)}{1 + 1} = \frac{40}{2} = 20 \quad (4)$$

To identify the popular content, let $X = \{x_1, x_2, \dots, x_n\}$ be the set contents and $W = \{w_1, w_2, \dots, w_n\}$ is corresponding weights which is equivalent to the Interest frequency such as $w(x) = I(x)$. If the content x_i has the Interests frequency $I(x_i)$ is greater than the threshold, the x_i is considered as popular. The popularity can be calculated using the following Eq. 5:

$$P_{x_i} = I(x_i) > DT \quad (5)$$

In the current above example (Eq. 4), the threshold is obtained as 20 and according to DCCC, the content x_i will be selected as popular if it has received a greater number of Interests as the threshold value. Therefore, according to the current scenario, the content x_1 is selected as popular and recommended to be cached at the network nodes.

Algorithm 2. is developed to identify the set of higher popular contents. It differentiates the content into sub-categories such as a set of high frequently Interested content and a set of low frequently Interested content based on the dynamic threshold value. Therefore, of the set of content $X = \{x_1, x_2, \dots, x_n\}$, the x_i is considered as low frequently Interested content and x_j is considered as high frequently Interested content where $x_i \wedge x_j \in X$. However, Algorithm 2, focuses on the highly frequently requested content and the content is shown as x_i in the given algorithm.

3.2. Content placement mechanism

As the content selection process gets complete the Content Placement Mechanism (CPM) is executed in which the selected popular content is cached at the selected network nodes according to the data caching algorithms. However, in the current strategy DCCC, the cluster-based caching mechanism is developed and content caching is done based on the cluster head selection.

Algorithm 2. Identify popular Content

Input: $X = \{x_1, x_2, \dots, x_n\}$ number of contents.
Output: Popular Content x_i
Procedure PopularContent (X, DT)

1. PopularContent = []
2. **for** x_i in X **do**
3. $I(x_i) = read_I(x_i)$ // Interests for x_i
4. $w(x_i) = I(x_i)$
5. **for** x_i in X **do**
6. **if** $w(x_i) > DT$ **then**
7. **return** PopularContent

3.2.1. Clustering

Clustering refers to the collection of network nodes concerning different properties and resources that form a cooperative caching system for load balancing, quality analysis, data processing, caching, and forwarding in advanced networks. A cluster consists of two components: the cluster head and cluster members. The cluster head is responsible for collecting content from the server, receiving Interests from subscribers and neighbor nodes, and forwarding them to the appropriate sources and subscribers. The cluster members are interconnected nodes connected to the cluster head or neighboring nodes.

3.2.2. Cluster head section

The cluster head selection is based on different factors like number of nodes connected to one node, distance from the provider, and residual energy. It improves network mobility and performance by selecting the cluster head among the nodes. As a result, the data delivery latency and energy consumption are reduced and cache hit performance is improved. Moreover, the cluster head selection depends on diverse algorithms, strategies, node power, node capacity, relative speed and node degree. Moreover, some mechanisms, is also depend on centrality nodes such as closeness centrality and betweenness centrality. The current study selects the cluster head based on the shortest distance from the end

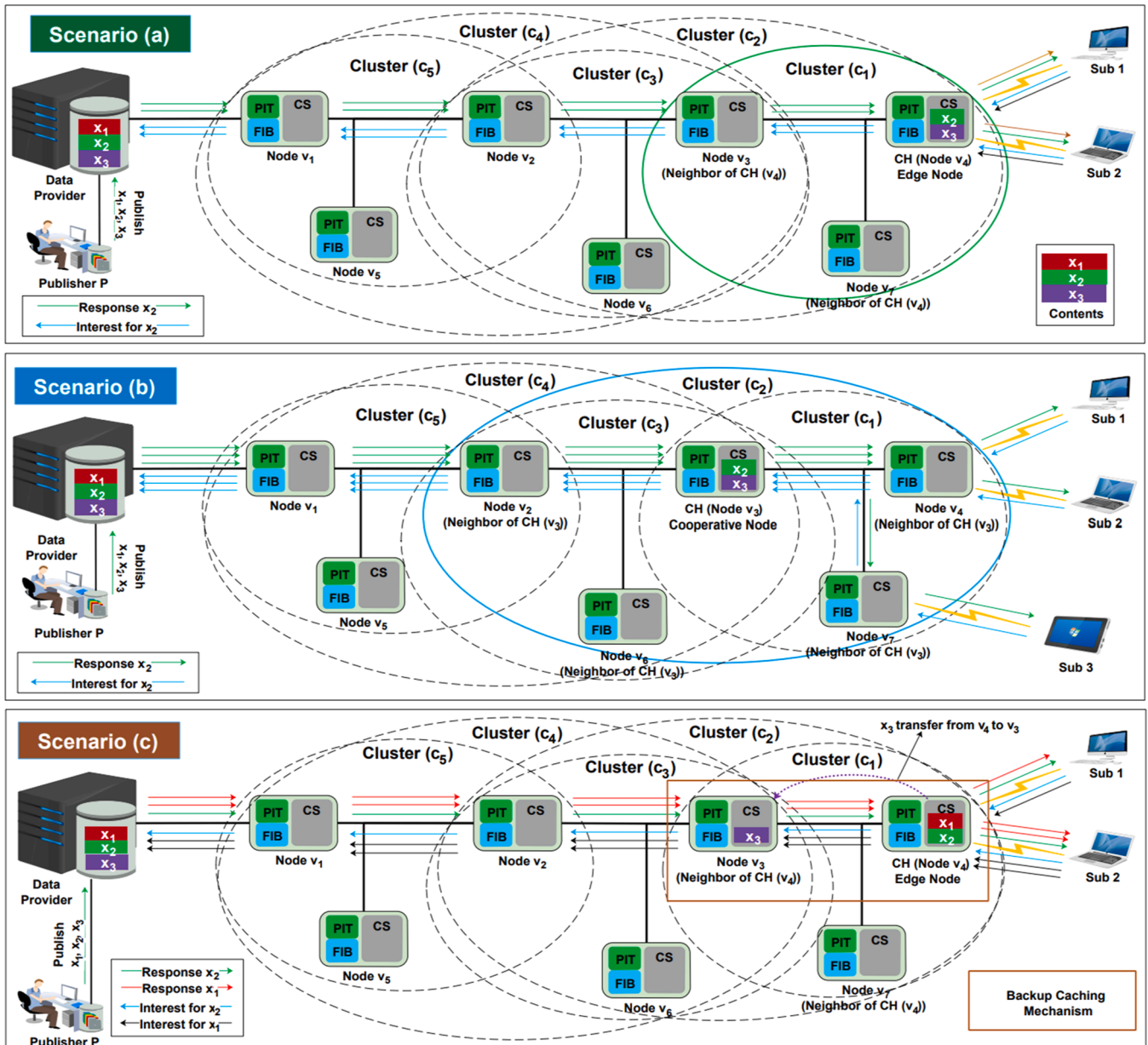


Fig. 1. Dynamic Clustering-based Cooperative Caching Scenario.

devices.

Algorithm 3. Clustering and Finding the Cluster Head.

Input:

$(G(V, E))$ a network is made up of nodes, edge, and subscribers.

Output:

Clustering: The set of cluster heads in a network.

Procedure Clusering($G(V, E)$):

```

1. CH = {} // Cluster Heads
2. Sub = {} //Subscribers
3. V = {} // Network Nodes
4. E = {} // Connections
5. dis = {}// Dictionary to cached distances
6. for each v in V do
7.   dis = mD (v, Sub) // mD measure Distance
8.   ve = min(dis) // ve edge node
9.   CH.add (ve)
10.  V.remove(ve)
11. while V is not empty do
12.  N = {} // Neighbors
13.  for each v in V do
14.    N = true
15.    for each ch in CH do
16.      if mD (v, CH) > mD(v, ch) then
17.        N = False
18.  break:
19.    if N = true then
20.      N.add(v)
21.      CH = CH ∪ N
22.      V = V - N
23.  return CH

```

Primarily, in DCCC, the edge node next to the subscribers is chosen as the cluster head and the connected neighbor nodes make the cluster. The reason is that we are proposing a dynamic cluster-based caching strategy in which the cluster is dynamically selected based on the Interests preference at a network node and after that, the cluster is created based on the cluster head. If a node shows a higher preference (higher numbers of Interests are received) of the subscribers, the node is selected as cluster head. However, the data delivery path may consist of multiple nodes that show similar Interests preference. Therefore, we already have

defined the condition of selecting cluster heads based on the shortest distance from the subscribers.

When we choose an edge node as cluster head, it delivers several benefits such as it minimizes the data delivery latency, and reducing the path stretch between caching node and subscribers to fulfill the demands of subsequent Interests. As a result, data availability and the cached ratio are increased and it reduces the inefficient usage of restricted cache memory. Therefore, most of the cache is used to accommodate the higher popular content. This ensures that popular content is accommodated at the most important nodes along the data delivery path, fulfilling the demands of a large number of Interests. This minimizes data retrieval delay and stretch, leading to a higher cache hit rate. Let $G(V, E)$ be a network in which the $V = \{v_1, v_2, \dots, v_n\}$ shows the number of network nodes while the $E = \{e_1, e_2, \dots, e_n\}$ represents the set of links or connections among the network nodes.

The $d(u, v)$ is the distance between two network nodes u and v and according to DCCC, the Edge Node (EN) is selected as cluster head and it can be defined as: $EN \in CH$ where $CH = \{ch_1, ch_2, \dots, ch_n\}$ is the set of cluster heads in a network and $CH \subseteq V$. Therefore, the edge node will be selected as cluster head if node meets the following conditions:

$$\forall u \in V, d(EN, u) \leq d(u, v) \quad (5)$$

where for all $v \in V, v \neq EN$. All the cluster heads are associated with a number of neighbor nodes and it can be defined as: let the $N(v)$ is the set of neighbor nodes of a node v or it can be said that $N(v)$ set of neighbor nodes directly connected to node v . Therefore, a cluster can be defined as: it is the collection of neighbor nodes and cluster head and it formed by joining the neighbor nodes with cluster head. Let $C = \{c_1, c_2, \dots, c_n\}$ be the set of clusters in a network and to find the i th cluster, the condition can be defined as:

$$c_i = \{ch_i\} \cup N(ch_i) \quad (6)$$

Fig. 1 shows two scenarios of DCCC, where the set of the nodes is given by $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and according to DCCC, the set of cluster is given as $C = \{c_1, c_2, c_3, c_4, c_5\}$. Therefore, according to Fig. 1 five clusters can be obtained dynamically regarding the subscribers' preference. Initially, the edge node (v_4) received a number of Interests for content (x_2) from subscribers (sub-1 and sub-2) as shown in Scenario (a). according to the DCCC, the Node (v_4) is selected as cluster head because, it has the shortest distance from the subscribers and all the Interests are received at Node (v_4). The Node (v_4) forwards the subscriber's Interests to download the content (x_2) from Data Provider. According to DCCC, the content (x_2) is selected as popular from the set of contents $X = \{x_1, x_2, x_3, \dots, x_n\}$. Therefore, the content x_2 is cached at the cluster head Node v_4 and the caching status of v_4 is shared with its neighbor node (v_3) and node (v_7). Thus, the first cluster (c_1) in Scenario (a) is shown by a solid green line and it is consisting of three nodes such as v_3, v_4 , and v_7 where v_4 is the cluster head while v_3 and v_7 are the cluster members. As a result, all the subsequent Interests for content x_2 are satisfied from cluster head Node v_4 . The cluster c_1 can be defined by the given equation:

$$c_1 = \{ch(v_4)\} \cup N(v_3, v_7) \quad (7)$$

Algorithm 3. illustrates the mechanism for creating a cluster using a collection of nodes. Furthermore, the cluster head is chosen based on the shortest distance from the subscribers who have already fetched the demanded content, and the neighbor nodes are selected based on the cluster head.

Algorithm 4. Content Freshness

```

procedure: dataFreshness(content, freshness)
TFS = 0 // total freshness score
ct = 0 // count
for x in X do
Tcurrent = getTcurrent // Tcurrent Current time
Lifecontent = Lifex
Tdifference = Tcurrent - Lifex
if Tdifference ≤ freshness then
TFS = 1 -  $\frac{T_{difference}}{freshness}$ 
TFS += FS // Freshness Score (FS)
if ct > 0 then
AFS =  $\frac{TFS}{ct}$  // Average Freshness Score (AFS)
else:
AFS = 0
return AFS

```

The novelty of the DCCC strategy depends on the distinctive integration of its several elements. Nevertheless, several cluster head selection mechanisms and algorithms are indeed available in the literature. However, our proposed caching strategy stands out due to its unique combination of several key factors. Primarily, our strategy takes into account the number of users' Interests which provides a more accurate prediction of ideal cluster head candidates based on Interests patterns. These predictive capabilities differentiate the proposed algorithm from the traditional algorithms that often depend on historical information. Moreover, our strategy incorporates dynamic network parameters and selects the cluster head based on the dynamic conditions of the network. These dynamic aspects differentiate it from the existing mechanisms. In addition, the proposed strategy not only considers traditional parameters like energy consumption and distance but also integrates factors related to node centrality and Interest preference. This comprehensive estimation of multiple criteria ensures a more balanced and optimized cluster head selection, a feature that may not be as prevalent in existing algorithms.

3.2.3. Content freshness

Each content has a specific lifetime, and once it expires, the content is discarded from the cache storage. The popularity of cached contents is calculated in the same way as it was before caching at the cluster head. To address concerns about caching irrelevant content, DCCC considers data freshness. Each content is assigned a specific freshness value by the provider at the time of publication. Once the freshness value expires, the content is evicted from the caching node. This ensures efficient use of cache storage. In NDN-based IoT scenarios, data freshness plays a crucial

role in real-time applications, contrasting with traditional Internet approaches. The caching status of the popular content at the cluster head is cooperatively shared with its neighbors within the cluster to inform them about the caching status. This facilitates effective management of the cache. The freshness value determines the content's life, indicating the period in which the content is considered valid and up-to-date. Therefore, from a set of freshness $F = \{f_1, f_2, \dots, f_n\}$ of the set of contents $X = \{x_1, x_2, \dots, x_n\}$ cached in a network, the f_i for a content x_i cached at a network node n_i can be defined by the Eq. 8:

$$f_i = t_i(x_i) \quad (8)$$

For any of the content x_i in v_i is about to expire the freshness become zero. Where $x_i \in X$ while $v_i \in V$ and $X = \{x_1, x_2, \dots, x_n\}$ shows the number of contents in a network node while $V = \{v_1, v_2, \dots, v_n\}$. Therefore, the freshness is given by Eq. 9:

$$f_i \leq 0 \quad (9)$$

The Eq. 9 shows when the life span of content is over the freshness becomes zero and the content is considered as expired. The algorithm (content freshness) defines the mechanism to find the content freshness value.

Algorithm 5. Find the Cooperative Node

```

Input: Cluster Head
Output: Cooperative Node
Procedure: FindCN (CH, V)
1. CN = null // Cooperative (CN)
2. for each v in V do
3.   ct = 0 // count
4.   for each ch in CH do
5.     if edgeExists (v, ch) then
6.       ct = ct + 1
7.       if ct > 1 then
8.         CN = v
9.         break
10.  return CN

```

3.2.4. Cooperative node selection

In this case, we will discuss the selection of the cluster head when incoming Interests are received from more than one subscriber. Consequently, when Interests are received from multiple subscribers, the cluster head is selected based on the shortest distance and the Cooperative Node (CN). The CN is the node where the multiple edge nodes are directly linked or connected. Basically, in DCCC, the dynamic clusters are obtained according to the frequency of Interests. If a node shows a higher Interest frequency, the node is selected as the cluster head, and the neighboring nodes become cluster members. Therefore, the content is also cached at the CN, and the CN will be selected as the cluster head, and the connected nodes become the cluster members. Thus, a copy of popular content will travel from the edge node to the CN gradually, considering the Interest frequency that appears at a node along the data downloading path.

Let the set of cluster heads such as $CH = \{ch_1, ch_2, \dots, ch_n\}$ and the

$V = \{v_1, v_2, \dots, v_n\}$ is the set of nodes in a network and the CN also belongs to the set of nodes such as $CN \in V$. Therefore, according to DCCC the CN is the node connected to the multiple cluster heads or edge nodes where the number of Interests was received from the multiple subscribers. Thus, the number of cluster heads from the set of $CH = \{ch_1, ch_2, \dots, ch_n\}$ connected to the CN can be defined as:

$$\exists ch_1, ch_2 \in CH \quad (10)$$

Moreover, to meet the second condition of CN the Interests should be generated from more than one subscriber such as Interests from the subscriber can be defined as $I_{pre} = \{I_{sub_1} > 0, I_{sub_2} > 0, \dots, I_{sub_n} > 0\}$. Therefore, a cluster heads cannot be selected as CN:

$$\forall ch \in CH \ \text{Ach} \neq CN \quad (11)$$

This equation guarantees the CN is a network node from the set of network nodes and it is connected with more than one edge node or cluster head nodes. Algorithm 5 describes the selection of a CN node in a network while the network receives Interests from multiple edge nodes of cluster head nodes. In scenario (b) in Fig. 1, the CN node selection is illustrated where three subscribers (Sub-1, Sub-2, and Sub-3) sent several Interests to download x_2 from Data Provider. Therefore, the content x_2 is considered as popular and let the x_2 be cached at Node v_3 . In this transmission, the Node v_3 is considered as the cluster head because it satisfies the second condition of choosing CN defined by DCCC. The Node v_3 fulfils both requirements such as the node should be selected based on the shortest distance from all subscribers and meets the requirements of the CN node. Therefore, the Node v_3 is selected as cluster head and the connected neighbors are by the following equation:

$$N(v_3) = \{v_2, v_4, v_6, v_7\} \quad (12)$$

where $N(v_3)$ represents the neighbors of v_3 . The whole cluster c_2 is denoted by the blue solid line in Fig. 1 (Scenario (b)). Therefore, the cluster c_2 can be defined as:

$$c_2 = \{ch(v_3)\} \cup N(v_2, v_4, v_6, v_7) \quad (13)$$

Therefore, the selected popular content x_2 cached at Node v_3 (CN) and its popularity is reinitiated with 1. Algorithm 5 defines the method to choose the CN as a cluster head for the caching of popular content x_2 . The popular contents that are selected are cached at the cluster head, and their popularity is reinitialized with a count of 1 in order to compare their popularity when new content will arrive to accommodate higher frequently Interested content at the most important location. When a cached popular content x_2 will receives more Interests the Interest count will be incremented. Therefore, the popularity of already cached content x_2 is calculated in the same way it was calculated before caching it at the cluster head. Algorithm 6 shows the initialization after the caching of popular content at the cluster head. Moreover, the popularity of cached content is measured and the content x_2 will be stayed based on the content freshness value or lifetime.

3.3. Backup caching mechanism

In the Backup Caching Mechanism (BCM), if the cache of the cluster head overflows and new popular content arrives to accommodate the contents at the cluster head, the popularities of the cached content are compared with the popularity of the incoming content. If the incoming content has greater popularity than the popularities of the cached contents, the incoming content is cached at the cluster head, and the content showing the lowest popularity is evicted from the cluster head and

cached at the upstream cluster member. The current popularity of the cached content is compared with the popularity of the incoming content.

Algorithm 6. Backup Caching Mechanism

Input:
 New Content
 Old Content
 Neighbor Node

Output:
 Content Caching Decisions

Procedure: CacheContent (NewContent)

1. **if** x_j not in CH_{Cache} **then**
2. **for** each $Cached_x$ in CH_{Cache} **do**
3. **if** $P(x_j) > P(x_i)$ **then**
4. Cache.add(x_j)
5. **if** $I_{count}[Cached_x] > 0$ **then**
6. move x_i to neighbour (cached x_i)
7. **break**
8. **else:**
9. move x_j to neighbor (cached x_j)

Each Interest is added with an extra field in which the interest count is calculated, and a popularity value is attached to the demanded content. If a new content, let's say x_j , arrives at the cluster head, and the cache of the cluster head is overflowed then the popularity of the new content x_j is compared with the

old content x_i , where x_i and x_j are elements of X. The content will be cached at cluster head based on two conditions as given by Eqs. 11 and 12:

$$\forall x_i, x_j \in X \quad (14)$$

$$P(x_j) > P(x_i) \quad (15)$$

If the new content x_j has a greater popularity value than the old content x_i , the new content x_j is cached at the cluster head node. As the new popular content x_j is cached, the count is reinitialized to zero, and it will be incremented as new Interests are received. Consequently, the efficient utilization of the cache in the cluster head node is important because it may happen that popular content will be cached at one node for a long time, and the new content may have slightly less popularity, but it should still be allowed to cache at the cluster head. All the contents at the cluster head node will be checked for popularity, and the content that shows the least popularity will be chosen for eviction and cached at a neighbor node within the cluster. The old popularity and current popularity are saved in a popularity table. Fig. 1, Scenario (c), illustrates the BCM in which several Interests are generated to fetch content x_1 from the Data Provider. Currently, according to Scenario (c) in Fig. 1, content x_1 has received three Interests from subscribers (Sub-1 and Sub-2), while content x_2 has received two Interests from subscribers (Sub-1 and Sub-2). According to DCCC, x_1 is considered the more popular content and is recommended to be cached at cluster head Node v_4 . Therefore, when the content x_1 arrives at cluster head node v_4 , the cache

of node v_4 overflows. Thus, the popularity of the arrived content x_1 is compared with the cached content x_2 and x_3 . As a result, content x_3 shows a lower popularity value, and it is evicted from cluster head node v_4 and cached at neighbor upstream node v_3 . Content x_1 is cached at cluster head node v_4 because of its higher popularity compared to contents x_2 and x_3 . Algorithm 6 shows the BCM for newly Interested contents.

4. Performance evaluation

The performance evaluation and experiments to check the effectiveness of the proposed DCCC NDN-based IoT caching strategy are conducted using ndnSIM which is an extension of Network Simulator 3 (NS3). We evaluate the proposed DCCC and compare strategies on a common platform to identify the performance of DCCC as compared to the earlier caching strategy with varying parameters. We compare our proposed caching strategy with four other strategies, such as Cache Everything Everywhere (CEE) [43], Hierarchical Cluster-based Caching (HCC) [44], Collaborative Filtering-based Content Caching (CFCC) [45] and Priority-based Content Popularity-Aware Caching (PCPA) [46].

4.1. Simulation settings and parameters

We used an Intel Core i7 machine with the following specifications: 16 GHz CPU and 8 GB of memory. The ndnSIM platform was chosen for the evaluation of the proposed DCCC with alternative caching strategies, as defined above. We selected a tree topology with five levels, where each parent node is connected to 0–3 child nodes, and the data provider is directly linked with the network core node at level one. The amount of content considered was 10,000, and each piece of content has the same size of 500 MB. Initially, all the content is cached at the data provider. The links among the intermediate nodes have capacities of 1 Gbps, while the links between edge nodes and subscribers have capacities of 100 Mbps. The poisson arrival process is chosen, and 20 Interests per second are disseminated to the edge node. The Zipfian content distribution law is modeled with a skewness value (α) chosen from 0.5 to 1.5. Table 2 shows the selected simulation parameters.

4.2. Performance metrics

The performance of the proposed DCCC caching strategy is evaluated in terms of Average Cache Hit Ratio (ACHR), Average Hop Reduction Ratio (AHRR), Server Hit Reduction Ratio (SHRR), and Content Retrieval Delay (CRD). The metrics and simulation experiments are given below:

Table 2
Evaluation Parameters.

Parameters	Values
System warmup time	50 s
Execution time	3550 s
Number of executions	20
Catalogue Size	10,000 contents
Interest Size	100 MB
Content Size	500 MB
CS size	10–50 contents
Interest rate λ	50 Interests/sec
Popularity model	Zipfian law
alpha parameter values	0.5–1.5
Total number of nodes	121
Data provider	1 node
Number of Subscribers	81
Tree topology	5 level
Child node	0–3
Caching node	39

4.2.1. Average cache hit ratio

ACHR is known as the typical parameter to evaluate the system performance of caching-based strategies, algorithms, and schemes. It is the ratio between the subscribers' Interests and the responses given by a cache that enables the network to the subscribers' Interests. The higher the value of ACHR the more efficient caching system is considered. It can be defined as:

$$ACHR = \frac{\sum_{i=1}^n I - count_r}{I_{total}} \quad (16)$$

where α shows the time cycle in which the hit rate is calculated, while $I - count_r$ represents the number of responses to the subscribers' Interests by caching nodes and I_{total} indicates the total number of Interests sent by the subscribers to the network.

4.2.2. Average hop reduction ratio

AHRR is the ratio between the number of hops that subscribers' Interest go through to fetch the demanded content by following a certain caching algorithm and the number of hops that subscribers' Interests to fetch content without caching nodes from the data provider or server. It is also known as the response distance of the path stretch. The AHRR will increase if the Interests follow the shortest path, and it can be measured as:

$$AHRR = \frac{\sum_{i=1}^n h - count_{cs}}{\sum_{i=1}^{|N|} h - count_s} \quad (17)$$

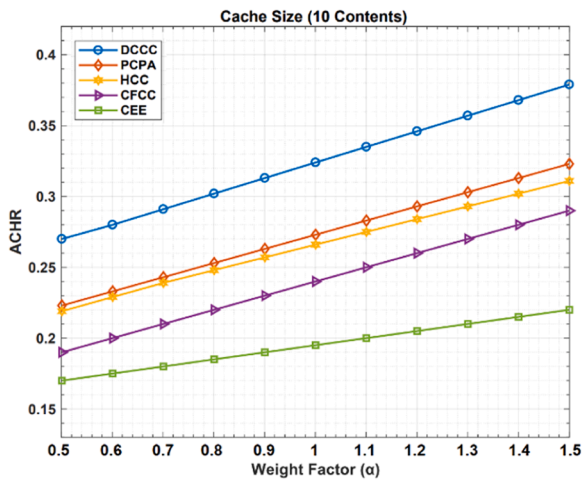
where $\sum_{i=1}^n h - count_{cs}$ represents the hops go through by the subscribers' Interests to fetch content from node cs , while $\sum_{i=1}^{|N|} h - count_s$ denotes that the Interests go through the number of hops to download contents from the server s .

4.2.3. Content retrieval delay

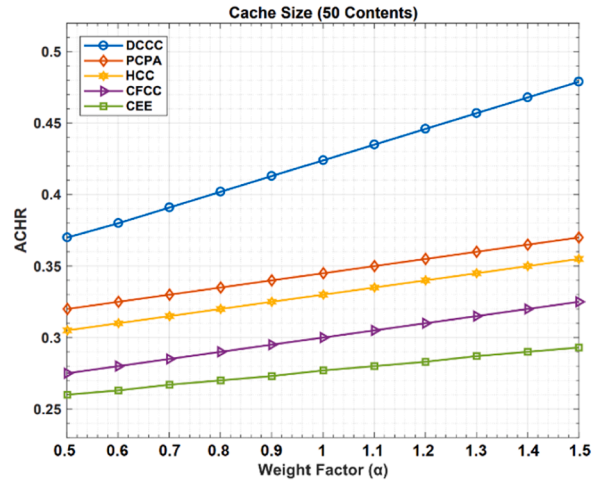
CRD represents the average number of time cycles in which a subscriber's Interests are sent to the caching node and the content is downloaded, or we can say the time required to fetch the content from the caching node. The shorter delay improves the caching performance of the caching network.

Table 3
Abbreviations and corresponding Description.

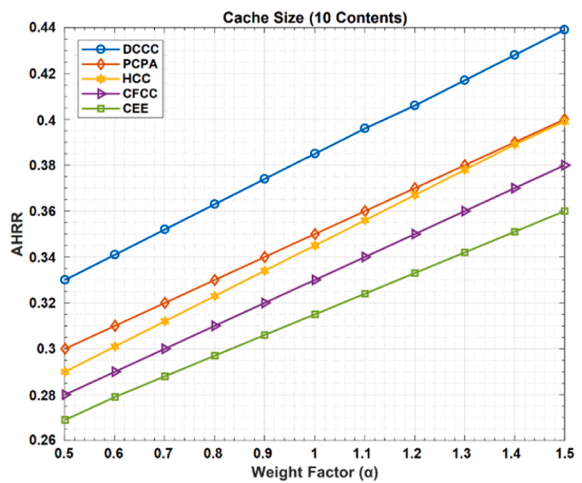
Abbreviations	Descriptions
NDN	Named Data Networking
IoT	Internet of Things
DCCC	Dynamic Clustering-based Cooperative Caching
CCC	Central Control Caching (
EPPC	Efficient Popularity-aware Probabilistic Caching
PoC	Popularity-aware Closeness
DCM	A. Data Collection Mechanism
DTPT	Dynamic Threshold-based Popularity Table
DCM	Data Collection Mechanism
CH	Cluster Head
WA	Weighted Average
PDT	Popular Data Table
CPM	Content Placement Mechanism
CN	Cooperative Node
BCM	Backup Caching Mechanism
ACHR	Average Cache Hit Ratio
AHRR	Average Hop Reduction Ratio
CRD	Content Retrieval Delay
CEE	Cache Everything Everywhere
HCC	Hierarchical Cluster-based Caching
CFCC	Collaborative Filtering-based Content Caching
PCPC	Priority-based Content Popularity-Aware Caching
QoS	Quality of Service



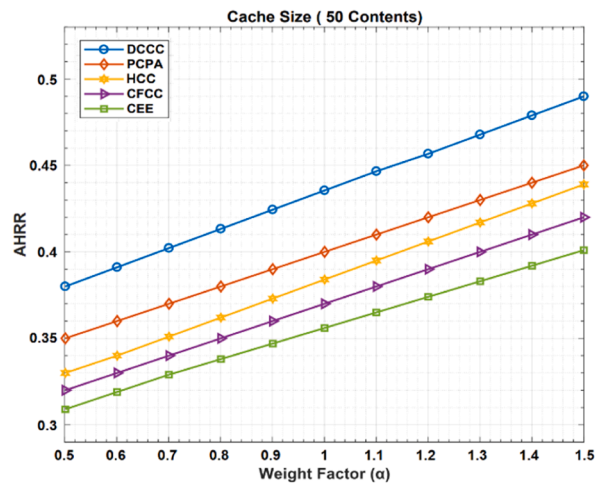
(a) Hit ratio on constant cache size (10 contents)



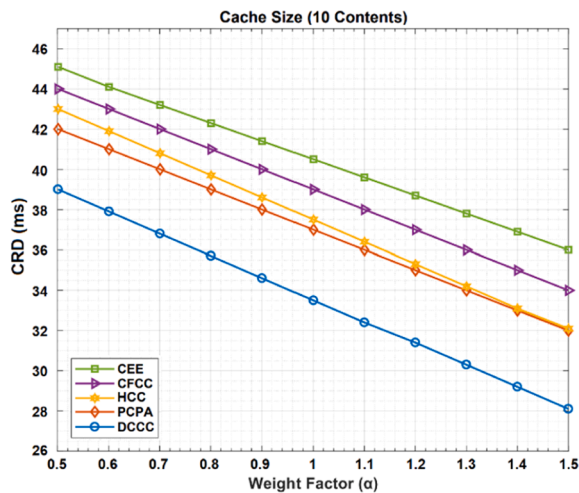
(b) Hit ratio on constant cache size (50 contents)



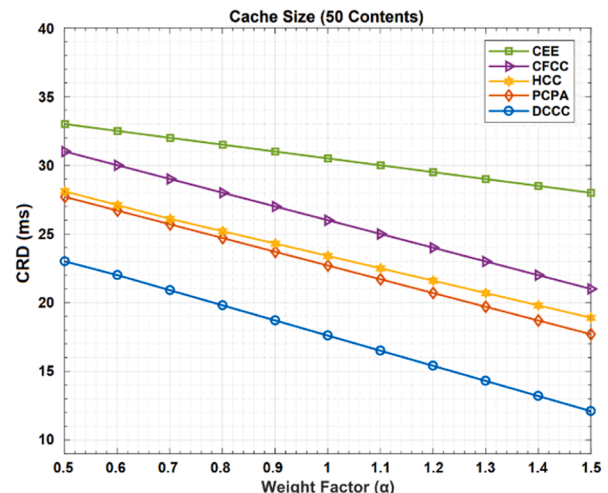
(c) Hop Count on Constant Cache Size (10 contents)



(d) Hop Count on Constant Cache Size (50 contents)



(e) Delay on Constant Cache Size (10 contents)



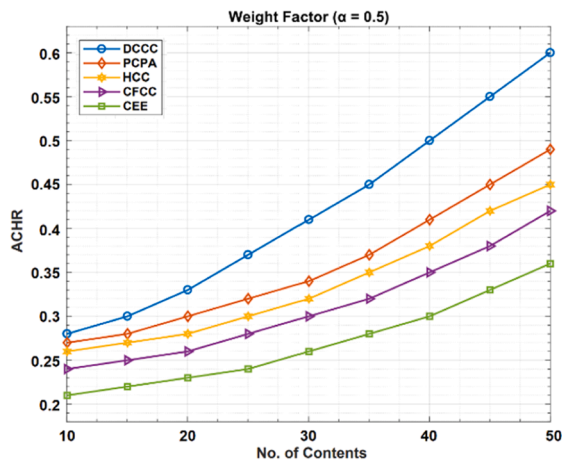
(f) Delay on Constant Cache Size (50 contents)

Fig. 2. Simulation on ACHR, AHRR, and CRD with Constant Cache Size.

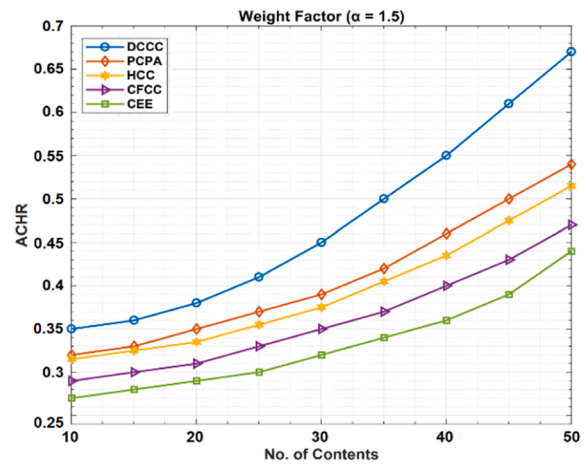
$$CRD = \sum_{t=1}^n D_I + \sum_{t=1}^n D_C \quad (18)$$

where $\sum_{t=1}^n D_I$ represents the delay in which the caching node receives

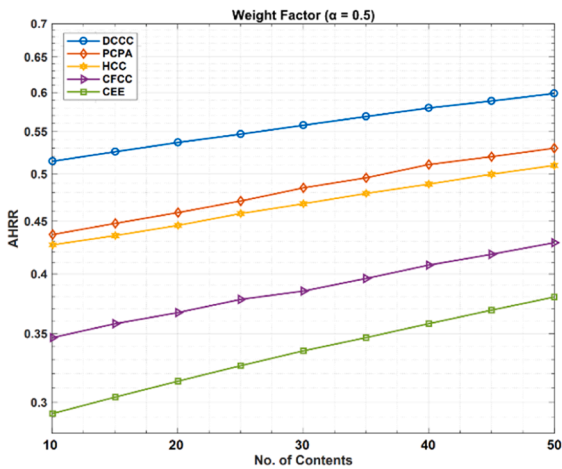
Interests and $\sum_{t=1}^n D_C$ shows the delay in which demand reaches the subscribers.



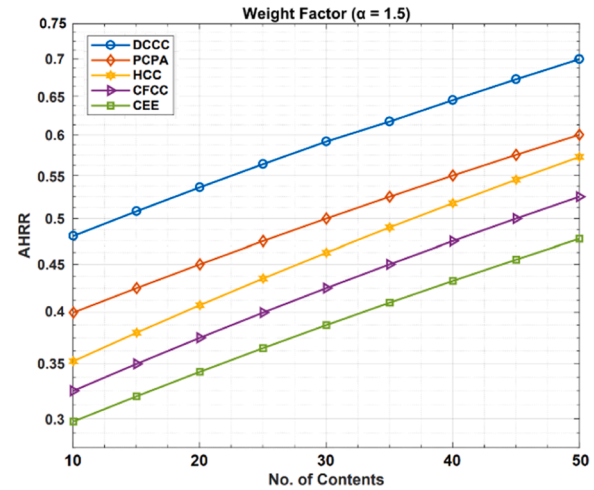
(a) Hit ratio on constant cache size (10 contents)



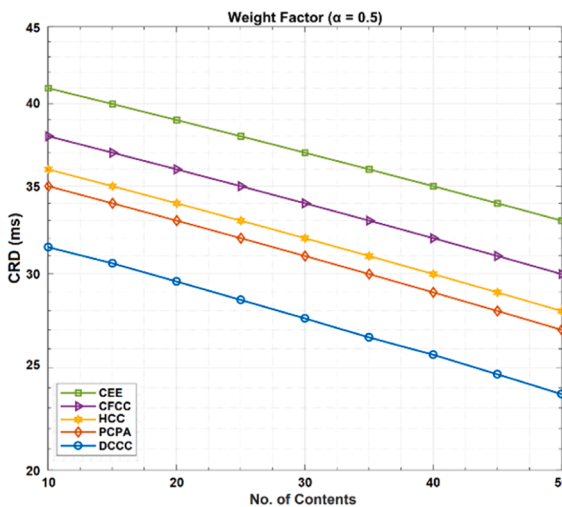
(b) Hit ratio on constant cache size (50 contents)



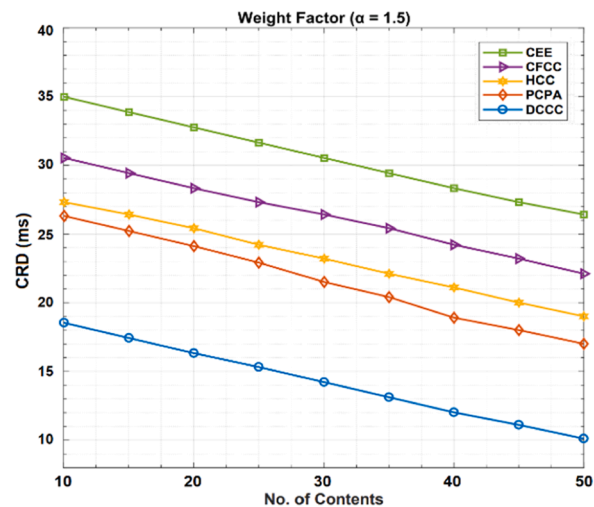
(c) Hop Count on Constant Cache Size (10 contents)



(d) Hop Count on Constant Cache Size (50 contents)



(e) Delay on Constant Cache Size (10 contents)



(f) Delay on Constant Cache Size (50 contents)

Fig. 3. Simulation on ACHR, AHRR, and CRD with Constant α Parameters.

5. Results and discussion

The evaluation is divided to present the experiment as defined in two phases based on different parameters. For the present study, we established an NDN-based IoT scenario, and the parameters are already defined in Table 3. Therefore, performance is measured based on ACHR, AHRR, and CRD to check the feasibility of the current proposed DCCC strategy. Constant and the cache size are taken as varying from 10 contents to 50 contents. The whole experiment is presented in two phases, where the cache size is taken as constant (10 contents and 50 contents) and the value of the weight factor is taken as varying from 0.5 to 1.5, while in the second phase the

value of the weight factor is taken as constant. Moreover, the average value of several executions is taken as the final result.

5.1. Phase I: constant cache size

We run tests on ACHR, AHRR, and CRD using the suggested caching strategy. We also compare the DCCC to four other strategies, including CEE, CFCC, HCC, and PCPA. Fig. 2(a, b) shows the simulation of cache hit performance by comparing strategies with the proposed one. The (c, d) shows the results on AHRR, while the (e, f) represents the outcomes of retrieval delay. Fig. 2 shows different performances with different constant cache sizes (10 contents and 50 contents). From Fig. 2, we can see that the performance of all caching strategies increases with an increasing α parameter. DCCC shows better cache hit, hop count, and retrieval delay results throughout the simulation process with both cache sizes and α parameters. The reason is that DCCC caches content according to the subscribers' Interest preferences at the network node where most of the Interests pass through to download content. In this way, the cache hit increases, and the delay and the number of hops to fetch the requested content get reduced. However, CEE performs poorly compared to other comparing strategies because it caches all the content at all the network nodes it goes through, increasing the unwanted cache usage of unpopular content. Therefore, for popular content, the incoming Interests need to traverse several hops to find the contents from the server, increasing the path stretch between the data provider and subscribers. As a result, delay and hop count are increased. Moreover, PCPA and HCC perform better than CEE and CFCC, and their performance increases relatively with different cache sizes and α parameters. CFCC shows better results than CEE because it caches contents in the cluster head situated near the end subscribers. However, all the caching strategies show better performance with a larger cache size (50 contents) compared to a smaller cache size (10 contents) because all the network nodes may accommodate more contents with a large cache size.

5.2. Phase II: weight factor α parameter

In this phase, the weight factor α is taken as constant, and its value is considered both maximum (1.5) and minimum (0.5), while the cache size is varied from 10 contents to 50 contents that can be cached at a network node. To check the effectiveness of the proposed DCCC model, we have divided our experiments into two different phases based on different caching sizes and weight factor α parameters. Fig. 3(a, b, c, d, e, and f) shows how well DCCC, PCPA, HCC, CFCC, and CEE work in terms of cache hit ratio, hop count reduction, and retrieval delay when cache sizes and content parameter values are changed. In the given figure, (a, b) shows the cache hit performance, (b, c) represents the performance on hop reduction ratio, and (e, f) demonstrates the performance on data retrieval delay. From all the results, it is clear that the performance of all comparing strategies is increasing as the cache size increases. However, the DCCC achieves better outcomes because of its structure of caching contents at the next node for the subscribers.

Moreover, it also provides an extra feature to cache content on cooperative nodes, from which most of the subscribers can easily fetch highly popular content within a short time interval. Additionally, it

implements a backup caching mechanism that allows accommodating slightly less popular contents at neighbor nodes rather than deleting them from the caching node. In this way, the cache hit performance becomes improved because all the incoming Interests are satisfied at the caching of the neighbor nodes rather than having to retrieve contents from the origin (server). So, the performance in terms of the hop reduction ratio gets better, and it takes less time to get popular content from neighbor nodes. CEE represents lower performance with all metrics and parameters. However, PCPA and HCC show better outcomes and perform relatively well because the contents are cached at the network edges according to both strategies. Therefore, both show relative performance. While CFCC shows lower performance because it executes several mechanisms like collaboration and clustering to perform one caching operation, Thus, the DCCC improves the overall network performance because it accommodates the demanded contents next to the subscriber nodes, fulfilling subsequent Interests within a short time. The reason is that DCCC provides an efficient content selection mechanism to select the optimal popular content and caches the selected popular content using a resourceful caching mechanism, placing it near the subscribers. This increases the cache hit and hop reduction performance while reducing the delay significantly.

6. Conclusion and future work

This research presents Dynamic Clustering-based Cooperative Caching (DCCC), a newly designed strategy that improves performance for IoT systems that use NDN. The DCCC model addresses key challenges in caching by incorporating three core mechanisms: Our caching system has three parts, such as DCM, CPM, and BCM, that work together efficiently. In DCM, we use a threshold value to find popular content, helping the system store and access it more effectively. CPM forms clusters by choosing nodes based on their connection to other nearby nodes and which content users want to access frequently, making it easier for servers to store and serve content that consumers need most often. With BCM, neighbors nearby in the network bundle cache servers to store moderately sought-after content, making this information easier to access and faster to load. The proposed model puts cache servers closer to where users live, which means data can reach them faster and they can find what they need more quickly, leading to better data retrieval performance. To validate the effectiveness of DCCC, we conducted extensive simulations using the NS3-based ndnSIM platform, comparing it against four existing caching strategies: We look at four different strategies in our study: CEE, HCC, CFCC, and PCPA. Our analysis showed that DCCC beat all other caching methods by showing better results in ACHR, AHRR, and CRD metrics while varying simulation parameters.

DCCC strategy brings remarkable improvements in caching in NDN-based IoT networks, but several valuable directions for the future are noted. Optimizing the strategy by recently implementing more sophisticated algorithms might expand the dynamic approach that would allow, in turn, identifying the most popular contents and applying, for instance, machine learning approaches to select the best caching decisions in real-time. Cooperating with other mobility models as well as using more complicated dynamic parameters, like changing network traffic intensity and nodes' failures, can improve the anti-robustness of the strategy in the actual world. Another field of interest is calculating the energy efficiency of cluster head selection and its consequences for sustainable IoT networks. Also, the improvement of the algorithms of content freshness concerning various applications or guaranteeing secure storing of data in cooperative caching can solve problems connected with privacy and reliability. These extensions may result in an adaptive, efficient, and secure caching framework for applications of the emerging IoT.

CRedit authorship contribution statement

Meng Yahui: Validation, Formal analysis. **Naeem Muhammad Ali:** Writing – original draft. **Bashir Ali Kashif:** Supervision.

Declaration of Competing Interest

In this article, we provide a broad view and insights regarding Named Data Networking-based Internet of Things (NDN-IoT) caching strategies. We discuss in detail how an NDN in-network caching feature can be a game changer for the IoT scenarios and how it can overcome the limitations of IoT, network analytics, and wireless sensor networks because of IP-based network system. This involved integrating survivability techniques with NDN principles, optimizing the caching strategy for improved content retrieval, and addressing network reliability challenges specific to IoT environments. To this end, we provide a comparative overview of the most recent studies and discuss various problems in IoT mechanisms with the objective to improve the overall network performance for IoT scenarios at the network edges. After that, we proposed an NDN-based IoT caching strategy to overcome the challenges of earlier caching strategies. We provide an extensive comparative performance analysis of NDN-IoT caching strategies. Through extensive simulations, we have observed that the NDN-IoT caching strategies can bring significant performance improvement in IoT-based environment due to its in-networking caching feature that in turn affect the content retrieval day and reduces the usage of resources in the network. We expect this study will open opportunities for a better understanding of Wireless Sensor Network (WSN) survivability through innovative approaches, including the application of Human-Inspired Deep Learning, network coding-based survivability, and addressing energy efficiency in real-time wireless networks. Finally, we confirm that the authors do not have any conflict regarding the paper contributions.

References

- [1] A. Morchid, Z. Oughannou, R.El Alami, H. Qjidaa, M.O. Jamil, H.M. Khalid, Integrated internet of things (IoT) solutions for early fire detection in smart agriculture, *Results Eng.* 24 (2024) 103392.
- [2] A. Morchid, I.G.M. Alblushi, H.M. Khalid, R. El Alami, S.R. Sitaramanan, S. Muyeen, High-technology agriculture system to enhance food security: a concept of smart irrigation system using Internet of Things and cloud computing, *J. Saudi Soc. Agric. Sci.* (2024).
- [3] Q. Wang, B. Lee, N. Murray, Y. Qiao, ECE: exactly once computation for collaborative edge in IoT using information centric networking, *IEEE Internet Things J.* (2023).
- [4] M.A. Naeem, Y.B. Zikria, R. Ali, U. Tariq, Y. Meng, A.K. Bashir, Cache in fog computing design, concepts, contributions, and security issues in machine learning prospective, *Digit. Commun. Netw.* (2022).
- [5] D. Doan Van, Q. Ai, In-network caching in information-centric networks for different applications: a survey, *Cogent Eng.* 10 (2023) 2210000.
- [6] L. Leira, M. Luis, S. Sargento, Context-based caching in mobile information-centric networks, *Comput. Commun.* 193 (2022) 214–223.
- [7] I.U. Din, S. Hassan, A. Almogren, F. Ayub, M. Guizani, PUC: Packet update caching for energy efficient IoT-based information-centric networking, *Future Gener. Comput. Syst.* 111 (2020) 634–643.
- [8] A.C. Castillo, An overview of information-centric network: concepts, network architecture, comparison, and difficulties, *Intell. Commun. Technol. Virtual Mob. Netw.* (2023) 27–42.
- [9] J. Quevedo, D. Corujo, Selective content retrieval in information-centric networking, *Sensors* 22 (2022) 8742.
- [10] A. Abrar, A.S.C.M. Arif, K.M. Zaini, A systematic analysis and review on producer mobility management in named data networks: research background and challenges, *Alex. Eng. J.* 69 (2023) 785–808.
- [11] A. Abrar, A.S.C.M. Arif, K.M. Zaini, M.H. Omar, Y. Meng, Advancing producer mobility management in named data networking: a comprehensive analytical model, *J. King Saud. Univ. -Comput. Inf. Sci.* 36 (2024) 102045.
- [12] Y. Meng, M.A. Naeem, M. Sohail, A.K. Bashir, R. Ali, Y.B. Zikria, Elastic caching solutions for content dissemination services of ip-based internet technologies prospective, *Multimed. Tools Appl.* 80 (2021) 16997–17022.
- [13] Z. Ali, M.A. Shah, A. Almogren, I. Ud Din, C. Maple, H.A. Khattak, Named data networking for efficient iot-based disaster management in a smart campus, *Sustainability* 12 (2020) 3088.
- [14] Y. Meng, M.A. Naeem, R. Ali, B.-S. Kim, EHCP: an efficient hybrid content placement strategy in named data network caching, *IEEE Access* 7 (2019) 155601–155611.
- [15] Y. Meng, M.A. Naeem, M. Sohail, A.K. Bashir, R. Ali, Y.B. Zikria, Elastic caching solutions for content dissemination services of ip-based internet technologies prospective, *Multimed. Tools Appl.* 80 (2021) 16997–17022.
- [16] M.A. Naeem, R. Ullah, Y. Meng, R. Ali, B.A. Lodhi, Caching content on the network layer: a performance analysis of caching schemes in ICN-based Internet of Things, *IEEE Internet Things J.* 9 (2021) 6477–6495.
- [17] R. Alubady, M. Salman, A.S. Mohamed, A review of modern caching strategies in named data network: overview, classification, and research directions, *Telecommun. Syst.* (2023) 1–46.
- [18] S. Alduayji, A. Belghith, A. Gazdar, S. Al-Ahmadi, PF-ClusterCache: popularity and freshness-aware collaborative cache clustering for named data networking of things, *Appl. Sci.* 12 (2022) 6706.
- [19] H. Al-Ward, C.K. Tan, W.H. Lim, Caching transient data in Information-Centric Internet-of-Things (IC-IoT) networks: a survey, *J. Netw. Comput. Appl.* (2022) 103491.
- [20] M.A. Naeem, R. Ali, B.-S. Kim, S.A. Nor, S. Hassan, A periodic caching strategy solution for the smart city in information-centric Internet of Things, *Sustainability* 10 (2018) 2576.
- [21] S. Deep, X. Zheng, A. Jolfaei, D. Yu, P. Ostovari, A. Kashif Bashir, A survey of security and privacy issues in the Internet of Things from the layered context, *Trans. Emerg. Telecommun. Technol.* 33 (2022) e3935.
- [22] R. Abbasi, A.K. Bashir, A.O. Almagrabi, M.B.B. Heyat, G. Yuan, Efficient lossless based secure communication in 6G Internet-of-Things environments, *Sustain. Energy Technol. Assess.* 57 (2023) 103218.
- [23] M.A. Naeem, M.A.U. Rehman, R. Ullah, B.-S. Kim, A comparative performance analysis of popularity-based caching strategies in named data networking, *IEEE Access* 8 (2020) 50057–50077.
- [24] S. Fayyaz, M.A.U. Rehman, M.S. ud Din, M.I. Biswas, A.K. Bashir, B.-S. Kim, Information-centric mobile networks: a survey, discussion, and future research directions, *IEEE Access* (2023).
- [25] M.A. Saleem, X. Li, K. Mahmood, S. Shamshad, M.F. Ayub, A.K. Bashir, M. Omar, Provably secure conditional-privacy access control protocol for intelligent customers-centric communication in VANET, *IEEE Trans. Consum. Electron.* (2023).
- [26] J. Li, J. Wu, C. Li, W. Yang, A.K. Bashir, J. Li, Y.D. Al-Otaibi, Information-centric wireless sensor networking scheme with water-depth-awareness content caching for underwater IoT, *IEEE Internet Things J.* 9 (2021) 858–867.
- [27] Q. Zhang, J. Wu, M. Zanella, W. Yang, A.K. Bashir, W. Fornaciari, Sema-IloVT: emergent semantic-based trustworthy information-centric fog system and testbed for intelligent internet of vehicles, *IEEE Consum. Electron. Mag.* 12 (2021) 70–79.
- [28] A.U. Rehman, Y. Alamoudi, H.M. Khalid, A. Morchid, S. Muyeen, A.Y. Abdelaziz, Smart agriculture technology: an integrated framework of renewable energy resources, IoT-based energy management, and precision robotics, *Clean. Energy Syst.* 9 (2024) 100132.
- [29] O. Hahm, E. Baccelli, T.C. Schmidt, M. Wahlisch, and C. Adjih, A named data network approach to energy efficiency in IoT, in 2016 IEEE Globecom Workshops (GC Wkshps), 2016, pp. 1–6.
- [30] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, S. AlAhmadi, Cache freshness in named data networking for the internet of things, *Comput. J.* 61 (2018) 1496–1511.
- [31] T. Miwa and S. Kimura, Cooperative update mechanism of cache update method based on content update dynamic queries for named data networking, in 2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW), 2019, pp. 33–39.
- [32] R. Shrimali, H. Shah, R. Chauhan, Proposed caching scheme for optimizing trade-off between freshness and energy consumption in name data networking based IoT, *Adv. Internet Things* 7 (2017) 11–24.
- [33] H. Asmat, F. Ullah, M. Zareei, A. Khan, E.M. Mohamed, Energy-efficient centrally controlled caching contents for information-centric internet of things, *IEEE Access* 8 (2020) 126358–126369.
- [34] M.A. Naeem, T.N. Nguyen, R. Ali, K. Cengiz, Y. Meng, T. Khurshaid, Hybrid cache management in IoT-based named data networking, *IEEE Internet Things J.* 9 (2021) 7140–7150.
- [35] N. Dinh, Y. Kim, An energy reward-based caching mechanism for information-centric Internet of Things, *Sensors* 22 (2022) 743.
- [36] M. Amadeo, C. Campolo, G. Ruggeri, A. Molinaro, Popularity-aware closeness based caching in NDN edge networks, *Sensors* 22 (2022) 3460.
- [37] N. Baltagiannis, I.A. Kapetanidou, and V. Tsaoussidis, EFPCCaching: Energy-aware and Fault-tolerant Probabilistic Caching of popular IoT content in ICN, in NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, 2023, pp. 1–4.
- [38] W. Huang, T. Song, Y. Yang, Y. Zhang, Cluster-based cooperative caching with mobility prediction in vehicular named data networking, *IEEE Access* 7 (2019) 23442–23458.
- [39] K. Devi Murugavel, P. Ramadass, R.K. Mahendran, A.A. Khan, M.A. Haq, S. Alharby, A. Alhussen, Maintaining effective node chain connectivity in the network with transmission power of self-arranged AdHoc routing in cluster scenario, *Electronics* 11 (2022) 2455.
- [40] M.A.R. Khan, S.N. Shavkatovich, B. Nagpal, A. Kumar, M.A. Haq, V.J. Tharini, S. Karupusamy, M.B. Alazzam, Optimizing hybrid metaheuristic algorithm with cluster head to improve performance metrics on the IoT, *Theor. Comput. Sci.* 927 (2022) 87–97.

- [41] A. Kumar, J.L. Webber, M.A. Haq, K.K. Gola, P. Singh, S. Karupusamy, M. B. Alazzam, Optimal cluster head selection for energy efficient wireless sensor network using hybrid competitive swarm optimization and harmony search algorithm, *Sustain. Energy Technol. Assess.* 52 (2022) 102243.
- [42] S.S.L.P.J. Shabu, K. Yadav, E. Kariri, K.K. Gola, M. AnulHaq, A. Kumar, Trajectory clustering and query processing analysis framework for knowledge discovery in cloud environment, *Expert Syst.* 40 (2023) e12968.
- [43] D. Gupta, S. Rani, S.H. Ahmed, R. Hussain, Caching policies in NDN-IoT architecture, *Integr. WSN IoT Smart Cities* (2020) 43–64.
- [44] H. Yan, D. Gao, W. Su, C.H. Foh, H. Zhang, A.V. Vasilakos, Caching strategy based on hierarchical cluster for named data networking, *IEEE Access* 5 (2017) 8433–8443.
- [45] D. Gupta, S. Rani, S.H. Ahmed, S. Verma, M.F. Ijaz, J. Shafi, Edge caching based on collaborative filtering for heterogeneous ICN-IoT applications, *Sensors* 21 (2021) 5491.
- [46] Y. Meng, A.B. Ahmad, Performance measurement through caching in named data networking based Internet of Things, *IEEE Access* (2023).