IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Performance Measurement through Caching in Named Data Networking based Internet of Things

Yahui Meng[1], Amran Bin Ahmad[2]

[1,] School of Science, Guangdong University of Petrochemical Technology, Maoming, China1; mengyahui@gdupt.edu.cn;

[1,2]InterNetworks Research Laboratory School of Computing (SOC) University Utara Malaysia 06010 UUM Sintok, Malaysia; amran@uum.edu.my;

*****Correspondence: Amran Bin Ahmad (amran@uum.edu.my);

ABSTRACT Named Data Networking (NDN) is considered the future of Internet architecture, providing a realistic solution for data delivery using a caching module in an Internet of Things (IoT) based environment. However, a major challenge of the caching module is data redundancy, which decreases the overall caching performance by caching similar data at numerous locations in an NDN-based IoT scenario. Moreover, the latency and stretch are maximized due to high redundant caching operations. Several attempts have been made by the research community to provide an enhanced solution to overcome such issues. However, the caching module still requires efficient enhancement. This study provides critical insights into earlier caching strategies. To solve the problems of these caching strategies, an enhanced caching strategy is proposed, named Priority-based Content Popularity-Aware (PCPA) Caching Strategy, which is evaluated by comparing its performance with some of the novel NDN-based IoT caching strategies. The proposed caching strategy outperforms the comparing strategies in terms of latency, hop count, cache hit ratio and energy consumption.

INDEX TERMS: Named Data Networking; Internet of Things; Caching

I. INTRODUCTION

The Internet of Things (IoT) comprises small and heterogeneous devices constrained by low power, low cost, and limited memory [1]. These devices, known as smart devices, include wireless sensors. Due to their limited memory and low power, data often becomes inaccessible. IoT-based applications such as smart towns, smart health, smart homes, and smart grids require enhanced privacy and security to retrieve data using these devices. Additionally, some IoT-based applications, such as smart transportation, require improved mobility services. End-users are primarily interested in downloading desired data rather than knowing the sources or physical locations of the data. For example, in Wireless Sensor Networks (WSNs), devices have a specific purpose of information harvesting [2]. Moreover, the IoT encompasses a multitude of small, low-power devices that have specific names or complex identification and address requirements for network identification. While IPv6 offers ample address space to address IoT device naming and addressing challenges, constrained devices struggle to handle long address spaces, resulting in increased resource consumption and communication difficulties [3, 4]. Currently, a vast number of IoT-based contents are generated and processed rapidly, and content may have multiple versions with different timestamps. This complexity in content naming increases the challenges of rapidly growing content. Therefore, it demands a highly reliable address space and permanent naming technique for both IoT-based heterogeneous devices and contents [5]. Moreover, interoperability and heterogeneity are critical aspects of IoT. The IoT network is built using billions of smart sensors to provide IoT services, and these sensors play a significant role in overall IoT-based communications [6]. However, these sensors (devices) are heterogeneous and constraint-oriented, having limited resources such as battery life, memory size, and processing power. Furthermore, communication is carried out among these sensors using several heterogeneous technologies such as wired, wireless, Bluetooth, cellular, Cognitive Radio Networks (CRN), and Long-Term Evolution (LTE) [7]. Therefore, IoT-based networks require high heterogeneity regarding device specifications and diverse communication techniques and technologies to ensure interoperability. In terms of data availability in IoT networks, data often becomes unavailable when a mobile node moves from one place to another [8]. Similarly, data cannot be forwarded when the device runs out of battery [9]. Therefore, techniques like in-network caching are highly required to improve data availability [10].

To address these critical challenges in IoT-based networks, Named Data Networking (NDN) is a promising

and flexible Internet architecture that provides all the necessary facilities to support IoT-based communications [11]. It is a challenging task for the current IP-based Internet architecture to meet the scalability demands of supporting billions of IoT devices and handling the enormous volume of data generated from such devices. In this regard, NDN provides a naming concept to efficiently support billions of devices by providing unique names and addresses to devices and data items, respectively [12]. Moreover, NDN introduces receiver-driven communication concepts to support IoT-based applications and offers flexible caching features to make data location-independent, enabling easy handoff for mobile devices [13]. Additionally, in NDN, the content is considered a self-certified entity that enhances the security and privacy of transmitting data for both the sender and the receiver [14]. Furthermore, named content makes it easy for the provider to verify that the data is disseminating according to the query flow, while location independence hides the source of the data. Moreover, the problem of heterogeneity among IoT-based devices can be easily solved by categorizing the devices using the NDN naming feature. Therefore, heterogeneous devices can operate with one another by integrating the NDN strategy layer into IoT-based networks [13]. This study focuses on the NDN-based caching feature that enables the IoT-based network to cache data anywhere and fetch similar data from nearby caching nodes with less delay, improving overall data availability and reducing the frequency of retrieving data from the provider. As a result, a significant number of network resources, such as energy, power, and network life, can be conserved [15, 16].

NDN-IoT caching provides several benefits, but it also imposes complications and restrictions on the development of caching strategies in different IoT networks. To propose a strategy for NDN-based IoT, it is necessary to consider certain properties related to the content and the node selected for content caching [17]. Content properties, such as freshness, popularity, timing ephemerality, provider, and node properties, such as the path stretch between the caching node and provider, battery (power level), and free memory, should be taken into account. Currently, researchers are interested in exploring content freshness, but content popularity is a more important parameter for enhancing caching performance, and measuring it requires urgent attention [18, 19]. Therefore, it is crucial to consider both content and node properties while making caching decisions. Additionally, content properties and node properties, such as Content Store (CS) size, battery life, caching module design, and node position in NDN-based IoT networks, should be taken into account. NDN-based caching fails to consider less memory, low power, and battery life, resulting in all incoming requests traversing several hops to find the required content, ultimately retrieving the content from the original provider [20, 21]. This maximizes the stretch ratio, resource consumption, content retrieval delay, and link utilization, reducing the data hit rate and overall caching performance of the network. Therefore, NDN caching strategies are more appropriate for improving IoT-based network performance. However, there is still a problem related to selecting an incentive node in a network that can provide efficient caching performance in multiple aspects, such as data availability, short latency, and lower energy consumption [22, 23]. Based on the aforementioned problems, we conducted a caching strategy in this study.

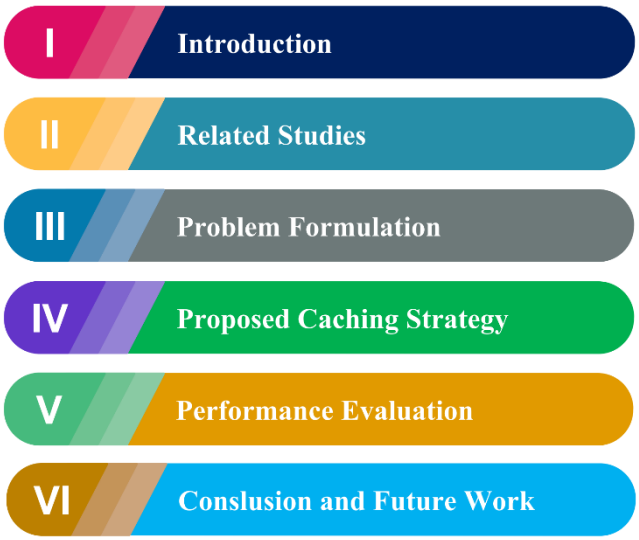

FIGURE 1. PAPER TEXONOMY

The contributions of the current study are presented as follow:

- This study provides a review of diverse caching concepts and issues in NDN-IoT caching environments. Primarily, we introduce the importance of NDN and IoT architecture and caching concepts.
- We define related caching strategies and identify problems with existing caching strategies.
- We critically categorize NDN and IoT-based caching strategies.
- We identify critical issues of NDN and IoT-based caching strategies.
- Based on the current issues of earlier NDN-based IoT caching strategies, we developed a new model to overcome these problems. We propose an NDN-based IoT caching strategy named Priority-based Content Popularity-Aware (PCPA) Caching.
- The proposed caching strategy provides three mechanisms: dynamic content selection, content caching, and content backup caching mechanism.
- According to the content selection mechanism, PCPA not only selects the most popular content based on the request frequency but also selects popular content by measuring the distance between end-users and caching node.
- The content caching mechanism provides the optimal caching node to cache the transmitted content and increases the cache hit performance.

- The backup caching mechanism provides the caching nodes to less popular contents to be cached near the end devices and reduces the data retrieval latency.
- We comprehensively compare the proposed strategy with different NDN-based IoT caching strategies, and we establish a simulation environment to check the efficiency of the proposed caching strategy.

The paper is organized as follows. Section II explains the problems based on the current study is proposed. Section III discusses the persisting issues with earlier studies that still need to be addressed. Section IV introduces the proposed caching strategy, while Section V outlines the performance evaluation process. Finally, the study concludes in the last section VI. Figure 1 show the paper taxonomy.

II. RELATED STUDIES

In NDN-based IoT networks, caching at intermediate nodes or devices offers several benefits, such as decoupling receivers or devices from the producers' location, reducing redundant data dissemination, and increasing the scalability of IoT networks [20, 24]. Moreover, caching can improve the energy consumption of IoT-based heterogeneous devices, and the overall network mobility can be organized more efficiently. Furthermore, employing caching features carefully can improve the life and flexibility of IoT-based networks. To improve caching performance, several NDN-IoT-based caching mechanisms (cache placement strategies and replacement policies) have been developed based on different criteria, such as probabilistic-based caching, centrality-based caching, content-based caching, node-based caching, and popularity-based caching [25, 26].

The NDN-IoT default caching strategy, called Cache Everything Everywhere (CEE) [27], provides a simple structure that offers high data availability and reachability. However, CEE can increase redundant content replications, leading to increased overall bandwidth and power consumption. Additionally, caching content at all network nodes can result in increased unwanted cache usage and unnecessary consumption of network resources. The Client Cache Strategy (CCS) [28] improves content validity by determining content popularity based on requests received at network nodes. However, this process can be time-consuming and reduce caching performance in terms of content retrieval latency and cache hit. CCS caches content at the betweenness centrality node, which can increase the overall stretch ratio for future requests and lead to link congestion.

The Tag-based Caching (TC) [29] strategy uses tag filters to look up and disseminate content, but managing tags can be challenging when a user applies the same tag to different content. This can result in increased cache storage consumption and reduced network and device performance. The Sleeping-based Strategy (SCS) [30] manages sleeping nodes to respond to incoming requests and improves energy performance. However, prioritizing content can increase caching and network congestion. The Periodic Caching Strategy (PCS) [31] uses Autonomous Systems (ASs) to cache content and improve system performance. However, content caching at the betweenness centrality node can become congested and lead to path stretch. Probabilistic Caching (ProbCache) [25] enhances content caching, freshness, and energy efficiency by using caching probability and dynamic attributes. It reduces content transmissions and caching operations, increasing cache hit rate and reducing content retrieval delay. The Cooperative Content Caching (CCC) [10] strategy improves energy efficiency and consumption by reducing the number of transmissions towards IoT devices. However, initiating several tables to execute upon receiving a request can increase communication overhead and reduce the data hit rate.

The Approximate Betweenness Centrality (ABC) [32] strategy improves caching performance under resource-constrained environments by determining centrality for each content. However, measuring approximate centrality can be challenging when the cache of a node becomes full and many requests are received for multiple contents. Probabilistic caching strategies have diverse mechanisms for determining caching positions, but performing multiple functions for each caching operation can consume more energy. Centrality-based caching strategies offer efficient caching performance by caching content at a centrality position. Hence, most of the subsequent Interests are satisfied from the centrality position, resulting in efficient caching performance. The centrality-based caching strategies promote the idea of content-centrism in which the disseminated contents are cached within the cache of the centrality node, such as the betweenness centrality node. The centrality node helps reduce latency because it caches the contents at the node linked with the maximum number of neighbor nodes. As a result, a large number of requests pass through the centrality node to obtain the content rather than traversing it to the main provider.

In probabilistic caching, the content's probabilistic value is measured to cache it at different locations to fulfill the demands of end devices. However, in popularity-based caching, the frequency of requested content is measured by taking the sum of incoming requests for that content. In content-based caching, IoT imposes several constraints on the content, such as freshness, probability, and popularity. In node-based caching, the appropriate node is required to be selected for caching the requested content [33, 34]. CEE stores contents by caching all the content at the routers, which increases the number of same-content replications. Leave Copy Down (LCD) [35] was developed to improve the content caching structure by reducing data redundancy. However, it increases the amount of similar data replications. Moving Copy Down (MCD) [36] was proposed to overcome the challenges arising from LCD and CEE strategies by removing the redundant data items in the data delivery path. The Edge Caching Strategy (ECS) [30] was developed to reduce the path length between the consumer and the cached content to satisfy end-users' demands. In proxy caching, the content retrieval time is minimized for subsequent Interests.

Table 1. Related studies and their basic aim and goals.

Sr	References	Aim/ Goal
1	Amadeo et al. [27]	It is known as default NDN-IoT caching strategy that provide simple caching structure and caches all the incoming content at all on-path caching nodes.
2	Wang et al. [28]	It caches incoming requested contents at betweenness centrality node.
3	Al-Ward H et al. [29]	It uses tag filters to look up and disseminate content for the caching of transmitted contents.
4	Amadeo et al. [30]	It manages sleeping nodes to respond to incoming requests and improves energy performance.
5	Naeem et al. [31]	It caches the incoming data items at the edge node and betweenness centrality node of each autonomous system.
6	Naeem et al. [25]	It defines probabilistic value at each caching node to decide the incoming content to be cached or not.
7	Arshad et al. [10]	This strategy improves energy efficiency and consumption by reducing the number of transmissions towards IoT devices.
8	Pfender et al. [32]	It improves caching performance under resource-constrained environments by determining centrality for each content.
9	Meng et al. [35]	It was developed the improve the structure of CEE caching strategy by reducing the redundancy.
10	Meng et al. [36]	It was developed to reduce the redundant data items at the data delivery path and improve the caching structure of LCD and CEE strategies.
11	Amadeo et al. [30]	In this caching strategy the incoming content are cached at the network edges to respond the subsequent user requests.
12	Hail et al. [37]	It evaluates the probabilistic values to identify the content caching node.
13	Naeem et al. [25]	It identifies the probabilistic values and forwards the caching contents hop by hop for most popular contents.
14	Barnardini et al. [38]	It caches the content using the threshold values to find the most popular contents.
15	Ren et al. [39]	It was developed using local gain and max-gain value to find the popular content and content caching nodes.
16	Current Study	It provides combination of mechanisms in which appropriate content selection, caching node selection, and backup caching mechanism are involved. It improves the content dissemination and data retrieval latency. It enhances the data availability and reduces the path stretch between end user and caching nodes.

However, it does not define any criteria to cache contents based on their popularity. Probabilistic caching provides several ways to cache content, such as Probabilistic Caching Strategy for Internet of Things (pCASTING) [37], random probabilistic caching, ProbCache, and Hope-based Probabilistic Cache (HPC) [25]. Similarly, HPC also improves homogeneous content replication at multiple locations. The Most Popular Cache (MPC) [38] was introduced to promote the most downloaded (popular) content. Max Gain In-network Caching (MAGIC) [39] was established to reduce bandwidth consumption and stretch (hop reduction). Therefore, the use of NDN caching strategies is appropriate for improving the performance of IoT-based networks. However, selecting the best incentive node to provide efficient caching performance remains a problem. The selection of popular content is crucial to improve caching performance, but it can also create complications and restrictions in developing caching strategies for different IoT networks. Therefore, an optimal caching strategy or combination of placement and replacement strategies is needed to enhance network performance in terms of data retrieval latency and energy consumption [21].

Table 1 shows the related studies and their basic goals. To overcome these challenges, a three-fold-based caching strategy named Priority-based Content Popularity-Aware (PCPA) caching strategy has been designed, which includes content selection, content caching, and backup caching mechanisms. The proposed caching strategy caches the content at the appropriate node along the data downloading path and enhances the performance of an NDN-based IoT network.

III. PROBLEM FORMULATION

According to the location dependency in the IP Internet, all Interests need to be forwarded to the remote server. As a result, the same content is traversed through several hops from the remote server to the consumer for each request [40]. This IP address-based communication increases the path stretch because the same content is transmitted from the remote server for each reply [41]. While NDN-IoT caching offers several benefits, it also presents complications and restrictions in developing caching strategies for different IoT networks. To propose an NDN-based IoT strategy, it is necessary to consider properties related to content and the node selected for content caching. Content properties include freshness, popularity, timing ephemerality, and provider, while node properties encompass the path stretch between the caching node and provider, battery level, and free memory.

Currently, researchers are keenly interested in exploring content freshness. However, content popularity is a much more important parameter for enhancing caching performance. Therefore, it requires urgent attention to measure the popularity of contents and improve the performance of the NDN-based IoT caching module. Moreover, it is essential to consider both content and node properties when making caching decisions. Content properties and node properties, such as CS size, battery life, caching module design, and node position in NDN-based IoT networks, should be taken into account. However, NDN-based caching lacks consideration for constraints such as limited memory, low power, and battery life. Consequently, when these conditions are not met, all incoming requests need to traverse several hops to find the required content, resulting in increased stretch ratio, resource consumption, content retrieval delay, and link utilization. This leads to a reduced data hit rate and overall caching performance of the network. Therefore, NDN caching strategies are more appropriate for improving IoT-based network performance.

However, there is still a problem related to the selection of an incentive node in a network that can provide efficient caching performance in multiple aspects, such as data availability, short latency, and lower energy consumption. Sometimes, content popularity has multiple effects, and when content has a higher popularity count but less usage in the future, efficient energy performance is reduced, consuming more resources and unproductive storage. This is because most frequently requested content cannot be accommodated in the node's cache, and a large number of incoming requests for that content are forwarded to remote providers, significantly increasing content retrieval delays. Centrality-based caching has a serious problem of selecting only a central node for content caching, which has limited cache storage, while a large number of contents are requested and cached. Consequently, it is impossible to cache all contents within the limited cache storage and improve overall caching performance. Therefore, it is highly required to select a node that can improve data availability for IoT-based end devices and network performance. On the other hand, selecting popular content is the most crucial part for improving caching performance. To achieve this, a dynamic mechanism is highly required to measure content popularities based on content requested frequency. Thus, IoT-based applications require contents with different characteristics, such as real-time applications needing fresh content, while flash crowds require contents with high requested frequency. Hence, it is considered that NDN-based IoT caching is still under construction and exists in its early stage

CEE has a simple structure that provides a caching-based environment with high data availability and reachability. However, it increases the redundant replication of contents, leading to higher overall bandwidth and power consumption. Additionally, it results in unwanted cache usage by caching undesirable contents at all nodes, which unnecessarily consumes network resources. To address these issues, pCASTING was proposed to enhance content caching mechanisms, content freshness, and energy efficiency. The caching procedure operates in a distributed manner based on caching probability and probabilistic decisions. It utilizes dynamic attributes to improve cache occupancy and energy efficiency, thereby reducing content transmissions and caching operations. Content caching probability is defined by measuring caching utility along with content freshness, giving higher freshness content a higher probability to be cached with a higher energy level. This approach increases the cache hit rate and reduces content retrieval delay. The Caching Fresh Popular Content (CFPC) [42] mechanism allows individual nodes to make independent caching decisions based on two primary factors: popularity count and content life. Popularity count is computed by aggregating user requests for specific content across all network nodes, while content life is determined by evaluating the freshness value considering its creation and expiration dates. As caching nodes become filled with content, new content must be accommodated to handle new requests. This cycle repeats every time a request is generated, resulting in most requests being fulfilled by the content publisher. To overcome these challenges, a caching strategy that enhances overall performance is necessary. Sometimes, content popularity can negatively impact efficient energy performance, as highly popular content may not be useful in the future, leading to increased resource consumption and unproductive storage. When frequently requested content cannot be accommodated in a node's cache, a large number of incoming requests for that content will be forwarded to remote providers, significantly increasing content retrieval delays.

The problems can be summarized as the need to efficiently enhance the content popularity parameters and mechanisms to improve data retrieval and cache hit performance. Additionally, the selection of nodes for content caching is a crucial measure in content caching decisions that affects performance when the data routing path becomes longer. Therefore, a node selection mechanism is highly required to enhance the overall NDN-based IoT network performance. It is also essential to select a node that

improves data availability for IoT-based end devices and network performance. Furthermore, selecting popular content is crucial for improving caching performance, and a dynamic mechanism is needed to measure content popularity based on the frequency of requests. IoT-based applications require content with different characteristics, such as real-time applications needing fresh content, while flash crowds require content with high request frequency. Therefore, it can be considered that NDN-based IoT caching is still under construction and in its early stages.

IV. PROPOSED CACHING STRATEGY

The objective of this study is to design a caching strategy that enhances the performance of NDN-based IoT networks. The proposed caching strategy aims to improve content selection and node selection mechanisms, as well as backup caching mechanisms. In NDN, caching modules play a significant role, with caching placement and replacement being crucial tasks for improving overall network performance. NDN caching strategies are more suitable for enhancing IoT-based network performance. However, there is still a problem concerning the selection of an efficient node in a network that can deliver optimal caching performance across various aspects, including data availability, short latency, and low energy consumption. Content popularity can sometimes have multiple effects, where highly popular content may become less useful in the future, resulting in reduced energy performance and increased resource and unproductive storage consumption. This issue arises because frequently requested content cannot always be accommodated in a node's cache, leading to a large number of incoming requests for such content being forwarded to remote providers, thereby significantly increasing content retrieval delays. While NDN-based IoT caching offers several benefits, it also presents complications and restrictions in the development of caching strategies for different IoT networks.

A. CACHING CONCEPT

In NDN-based IoT, content selection is based on users' request frequency, which indicates the number of requests received for a specific content. To determine content popularity, the number of requests for that content must be identified. This involves identifying the nodes and paths used to disseminate the requests and retrieve the desired content. All nodes in the NDN-based IoT network can measure the number of incoming requests for a particular content. The content with the highest request frequency value is considered to be the most popular content. Thus, it has a higher probability of being selected as popular content, also known as high popular content. Therefore, if the data contents are cached or transmitted through these network nodes and sensed by the nodes, they can be measured using the following equation:

$$\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \dots \dots \dots \boldsymbol{x_n}\} \tag{1}$$

where $x \in X$ and $|X|$ indicates the total number of contents following the network path from publisher to subscriber. Since the content is divided into chunks and all the chunks are considered the same size, the total number of contents can be calculated using the following equation 2:

$$X = \sum_{c=1}^{n} x_i \tag{2}$$

where $c_i \in C$ and $|C|$ shows the total number of contents.

B. CONTENT POPULARITY

Content popularity can be identifying through probability matrix. It can be measured using the following equation 3 and 4:

$$P = \{p_1, p_2, p_3, \dots \dots \dots p_n\} \tag{3}$$

$$P = \sum_{p=1}^{n} p_i \tag{4}$$

where $p_i \in P$ and $|P|$ shows that magnitude of popularity. For a specific content x the popularity can be measured using the following equation 6:

$$P_x = \frac{r_x}{R_T} \tag{5}$$

where r_x shows the number of requests generated to download content x while R_T represents the number of requests generated for all contents cached at the network nodes. The popularity of a specific content at time t can be calculated by the equation 7 given in the following:

$$P_x(t) = \frac{r_x}{R_T}(t) \tag{6}$$

If the content x already cached at a network node n then the following equation 7 will be defined the function:

$$P_x^n(t) = \frac{r_x^n}{R_T^n}(t) \tag{7}$$

where r_x^n shows the number of requests received for content x and node n at time t while R_T^n represents the total number of requests received at node n for all contents. We considered an NDN-based IoT environment in which IoT-based sensor nodes or IoT devices are used to generate IoT-based contents and called publisher or content producers. As traditional practices, the NDN-based IoT nodes will remain in sleeping mode until they receive a user request or an event happen. As a request reaches at nodes, the nodes get activated to

respond the incoming request by sending the requested content or it needs to forward the incoming requests. The number of nodes between publisher and subscriber can be denoted using the following equation:

$$|N| = \{n_1, n_2, n_3, \dots \dots \dots n_i\} \quad (8)$$

where $|N|$ shows the total number of network nodes. It can be denoted as:

$$|N| = \sum_{n=1}^{n} n_i \quad (9)$$

where $\boldsymbol{n_i} \in \boldsymbol{N}$ and $|\boldsymbol{N}|$ shows the total number of nodes.

C. CONTENT FRESHNESS

Unlike traditional Internet data, in the NDN-based IoT caching scenario, the lifespan of data plays a crucial role in implementing real applications in an IoT environment. Each content in the NDN-based IoT is associated with a specific time period assigned by the publisher or data producer, and once that time period elapses, the content is considered expired. In IoT applications, this time period is referred to as the content freshness value or content lifetime [43]. Freshness indicates the lifespan of a content and indicates how recently the content was generated by the publisher. Caching IoT-based contents is more challenging compared to traditional Internet contents because IoT caching decisions need to consider freshness or content lifespan, which depends on the type of content. The freshness may vary for different applications with the same content; for example, some applications require highly fresh data, while others can be satisfied with older data [44]. The content freshness can be defined using the following equations:

$$F = \{f_1, f_2, f_3, \dots \dots \dots f_n\} \quad (10)$$

$$F = \sum_{f=1}^{n} f_i \quad (11)$$

where $f_i \in F$ and $|F|$ shows the maximum freshness. In order to calculate the freshness for a specific content the following equation 12 is specified the function to measure the freshness:

$$F_x = T_x^c - T_x^g \quad (12)$$

where the F_x shows the freshness of content x, T_x^c demonstrates the current time and the T_x^g shows the time when the content X was generated. Therefore, the freshness can be defined as the time deference between current time and the time of content generation.

D. INTERMEDIATE NETWORK NODE

The NDN-based IoT network consists of wireless sensor nodes or IoT devices that have the capability to identify, process, and cache transmitted contents. When users send requests to fetch content from the producer, these requests follow a path and pass through multiple network nodes to reach the producer. The nodes responsible for transmitting the user's request to the producer are referred to as intermediate network nodes. Therefore, all the nodes between the user and the producer are considered intermediate nodes. Most of the time, NDN-based IoT nodes remain in sleeping mode until an event occurs, such as a request reaching the node to fetch some content. When the event occurs, the nodes get activated to perform the required sensing task and send the requested content back to the end user. For simplicity, let's assume that all NDN-based IoT caching nodes can sense one type of content. If a user sends out requests, these requests can be measured using the following equations. In NDN, the users' requests are commonly referred to as "Interest," so the users' Interests can be defined as:

$$R = \{r_1, r_2, r_3 \dots\dots, r_n\} \quad (13)$$

$$R = \sum_{r=1}^{n} r_i \quad (14)$$

where $r_i \in R$ and $|R|$ shows the maximum number of users' requests. If the number of requests were generated at time t then the equation 13 and 14 will defined as15 and 16 given in the following:

$$R(t) = \{r_1(t), r_2(t), r_3(t), \dots\dots, r_n(t)\} \quad (15)$$

$$R(t) = \sum_{r=1}^{n} r_i(t) \quad (16)$$

As the number of Interest (request) are received for content x at network node n at time t then the equation 16 will be converted in to the following 17:

$$R_x^n(t) = \sum_{r=1}^{n} r_i(t) \quad (17)$$

$$r(t) = x, n, f, (t) \quad (18)$$

where $r \in R$, $f \in F$, $n \in N$, and $x \in X$. Equation 18 shows that each request r generated for content x having freshness f and t is the time at which the Interest reached at a node n. The NDN-based IoT network nodes are associated with the storage capabilities to cache the transmitted contents for fulfill the user demands by sending the content from the caching nodes to the end users. Therefore, the caching nodes

between user and publisher can be defined by the following equation 19:

$$|C| = \{c_1, c_2, c_3, \dots\dots, c_n\} \quad (19)$$

$|C|$ is the amount of cache storage of the network and if the content x is cached at the network nodes as a traditional prospective in NDN a copy of content is cached at all the nodes between user and publisher:

$$|C| = \{c_{x_1}, c_{x_2}, c_{x_3}, \dots\dots, c_{x_n}\} \quad (20)$$

When the content x is cached at network node n at time t then the equation 20 will be converted into equation 21:

$$C(t) = \{c_{x_1}(n,t), c_{x_2}(n,t), c_{x_3}(n,t), \dots\dots, c_{x_n}(n,t)\} \quad (21)$$

According to the current study the same content x will not be cached at all the network nodes between user and publisher. Therefore, in order to show the current caching status regarding the present study, the equation 22 shows the content x will be cached at node i at time t.

$$C(n,t) = \sum_{c=1}^{n} c_{x_i}(n,t) \quad (22)$$

where $c_i \in C$. The caching status of a network node can be defined suing the binary values (0, 1). If the content is cached at node the status will be 1 otherwise, it will be 0. Therefore, if the content x cached at node n at time t the status will be shown by the following equation 23:

$$c_{x_i}(n,t) = 1 \quad (23)$$

If the content x cannot found at node n at time t the caching status is shown by the following equation 24:

$$c_{x_i}(n,t) = 0 \quad (24)$$

$$\sum_{c=1}^{n} S.c_{x_i}(n,t) = d \quad (25)$$

where S shows the size of data content x cached at ith node and d represents the size of cache of intermediate node i. Therefore, in order to find the caching status at any-time along the data delivery path at time t and ith node using the following decision strategy:

$$Strategy = C(n,t) \rightarrow C(n,t+1) \quad (26)$$

According to the equation 26, we can obtain the new caching status of an intermediate network node at time $t+1$.

A. SYSTEM MODEL

To propose a new strategy for NDN-based IoT, it is necessary to consider certain properties related to content and the nodes selected for content caching. These properties include content freshness, content popularity, provider information, as well as node properties such as path stretch between caching node and provider, battery power level, and available memory. The selection of popular content is crucial for improving caching performance. Therefore, a dynamic mechanism is required to measure content popularity based on request frequency. IoT-based applications require content with different characteristics. For example, real-time applications require fresh content, while flash crowds demand content with high request frequency. This suggests that NDN-based IoT caching is still under development and in its early stages. Caching plays a significant role in enhancing the overall performance of an NDN-IoT network. By caching appropriate contents along the data downloading path, caching strategies can efficiently meet end-user demands in a short time. To further enhance network performance, it is essential to design an optimal caching strategy or a combination of placement and replacement strategies. This strategy should aim to improve data retrieval latency, energy consumption, hop count, and cache hit ratio. In this study, a threefold-based caching strategy called Priority-based Content Popularity-Aware (PCPA) caching strategy is proposed. It combines three mechanisms: content selection, content caching, and backup caching. The content selection mechanism in PCPA is responsible for selecting specific content in the proposed caching strategy. Unlike other content popularity-based caching strategies, the proposed strategy selects content based on both the frequency of user requests and the number of hops traveled. This is achieved by measuring the number of incoming requests for a particular content in recent time and evaluating the distance traveled by multiple requests to fetch the same content. All NDN-based IoT network nodes can measure the number of incoming requests for a specific content using the Pending Interest Table (PIT).

The content with the highest request frequency value is considered as the Highly Frequently Requested (HFR) content and has a higher probability of being selected. However, in this strategy, the content cannot be selected as HFR content based solely on request frequency. The number of hops traveled by user requests is also taken into account. The popularity evaluation considers the number of hops traveled by each request to measure the distance between the end-user and the NDN-based IoT caching node. By measuring the distance, requests that have traveled a greater number of hops are identified, and the Average Hop Distance (AHD) is determined.

Therefore, a content has a higher chance of being selected as HFR if its average distance traveled by requests is higher than other contents. This is because some requests may reach the provider node after traveling a large number of hops, while others are sent from nodes located near the provider node or one hop away. As a result, requests sent

from closer nodes can reach the provider node earlier than those sent from farther nodes. In general, content is selected to be cached regardless of where the requests originate. This leads to a larger number of redundant requests being transmitted to download the same content multiple times from the distant provider node, increasing overall content retrieval latency and the average number of hops.

Algorithm 1. Content Freshness

Input:
Incoming requests
Output:
Fresh contents
***Procedure**:* GetFreshContent
1: user request (r_i) reaches at node n_i
2: ***for*** (check the node n_i has the requested content x_i)
3: ***if*** (required content x_i found at node n_i) ***then***
// According to equation 12
4: ***if*** ($T_{Current} > x_i.T_{Expire}$) ***then***
5: forwards the user request (r_i) to the pub
6: ***if***(publisher returned fresh content x_i) ***then***
7: refresh content x_i at node n_i
8: ***else***
9: evict the expired copy of content x_i from node n_i
10: ***end if***
11: ***end if***
12: ***end if***
13: ***end for***
14: ***return*** fresh copy of content x_i
[where $i = \{1, 2, 3, \dots \dots n\}$, and $x_i \in X,\ r_i \in R$]

To address this issue, the proposed caching strategy takes into account the average distance as a primary consideration. It selects content for caching based on both HFR and AHD. However, a question arises: what happens if a content meets the first condition of being HFR (receiving a greater number of requests than other contents) but fails to meet the second condition of AHD? In such cases, the content freshness or content lifetime is considered to choose an appropriate content as popular. In an NDN-based IoT scenario, content freshness or content life is crucial for improving overall network performance. Content providers set the freshness or content lifetime and attach it to the content header. When the content life expires, the content is considered invalid and evicted from the node to make room for new contents. The algorithm (Algorithm 1) identifies the content freshness. In some cases, the content producer may need to create a new copy of the expired content. IoT applications often require fresh contents, such as healthcare services that need up-to-date information to make appropriate decisions based on vital parameters like blood oxygen level and blood pressure. Therefore, if a content does not meet both conditions mentioned earlier, its content freshness or content lifetime is considered to determine the appropriate popular content for caching. The content with a higher lifetime contains highly fresh information, indicating its popularity in recent time. Thus, content lifetime is a crucial factor in choosing a suitable content as popular. The content selection algorithm (Algorithm 2) demonstrates the selection of highly fresh content.

Algorithm 2. Content Selection Mechanism

Input:
Incoming requests
Incoming requests for a specific content
Output:
Most frequently requested contents
***Procedure** GetPopularContent*:
1: multiple interests arrive at intermediate node n_i
2: ***for*** (requests r_i checks whether it has required content x_i)
3: ***if*** ($x_i.r_iCount > x_j.r_jCount$ or $x_i.n_i$Count > $x_j.n_j$Count) ***then***
4: ***if*** ($x_i.f_i > x_j.f_j$) ***then***
7: select x_i
8: ***else***
9: forward r_i the to the next node n_j
10: ***end if***
11: ***end if***
12: ***end for***
13: ***return*** most frequently requested content
[where $i, j = \{1, 2, 3, \dots \dots n\}$, $i \neq j$ and $x_i, x_j \in X,\ r_i, r_j \in R,\ f_i, f_j \in F$]

In the first phase, the content selection mechanism is discussed, where users' requests are measured and the distance for each request is calculated to determine the average distance for choosing popular content. Once the content selection mechanism is completed, the execution process moves to the next phase, which is the content caching mechanism.

In the content caching mechanism, caching decisions are made based on the content selection. These caching decisions occur along the data delivery path from the content producer to the requested users. Caching the content can typically reduce data retrieval latency and path stretch for subsequent requests from end users. Efficient caching decisions require the computation of several parameters, including path distance, caching nodes, disseminated content, number of hops along the path, and available space in caching nodes. Different content caching mechanisms have been designed, such as probabilistic caching, single node caching, and centrality-aware caching.

On the other hand, the Priority-based Content Popularity-Aware (PCPA) strategy provides a threshold value to assign to each node for identifying the content caching node. The content will be cached on the node that exhibits more interests than the threshold value. This threshold is determined by calculating the predictor history and incoming users' requests to identify the caching node. When a number of requests are generated to fetch a specific content from the content provider, the requests propagate from the user through various network nodes until they reach the content provider node. Each network node creates an entry as the request passes through it, and when the requested content is found, the node sends the content back to the user.

Furthermore, the request entry is deleted from each node as the content is disseminated through the nodes. According to PCPA, each node is associated with a static threshold value.

Algorithm 3. Content Caching Mechanism

Input:
Demanded Content: Content which is needed by user
Output:
Content cached at intermediate node
Procedure: GetCachingNode
1: content x_i reached at node n_i
2: ***for*** (check node n_i has content x_i)
3: ***if*** (node n_i has content x_i) ***then***
4: forward content x_i to next node x_j
5: ***else if*** (node n_i has enough cache space) ***then***
9: ***else if*** (x_i.node n_i has $r_i > 2$) *then*
10: ***else if*** (neighbors.node $n_i > 2$) ***then***
11: cache content x_i at node n_i
12: ***end if***
13: ***end else if***
14: ***end else if***
15: ***end else if***
16: ***end for***
17: ***return*** content x_i cached at node n_i
[where $i, j = \{1, 2, 3, \dots \dots n\}$, $i \neq j$ and $x_i \in X, x_j \in X$]

The threshold is assigned as $N_{threshold} > 2$ where $N = \{n_1, n_2, n_3, \dots \dots \dots n_n\}$. Therefore, each node has a specific count which means if a node receives more than two requests the node will be recommended to be cached that content for which the requests are received. More specifically, if a node receives more than two requests for a specific content the node will be selected to be cached that content. Here, a question is raised why dynamic threshold is node selected for the content caching? If the dynamic threshold is used the dynamic value will be increased with the time and a time will come when the incoming content show the smaller number of requests even it is selected as popular but the node dynamic value is greater than value node has received number of requests. As a result, the content will not be cached at any of the intermediate nodes. For example, at node N a content A is cached recently and the dynamic threshold is calculated as 10 at that time to cache content A. After a small time, a content B is arrived at node N and node N received 8 requests for content B. According to the dynamic threshold value 10, the content B will not recommend to be cached at node N. Thus, the dynamic threshold value will be increased as the increasing in number of requests and it will not appropriate to caches the contents showing less popularity than recent contents. Hence, the overall performance is reduced. Moreover, it is not only the condition to cache the content at a network node. However, in order to reduce the redundant replications of same content, the node should be connected with more than two neighbor nodes or the node should have more than two outgoing interfaces. Therefore, if the node connected with more neighbors have more probability to caches the content. Usually, all the network nodes are associated with two neighbors in which one is incoming and one is out going. Therefore, if all the network nodes will allow caching all the content, then the proposed model will behave like NDN default caching strategy EEC and it increases the content redundancy.

Algorithm 4. Back up Caching Mechanism

Input:
Demanded Content: Content which is needed by end user
Output:
Content cached at back up caching edge node
Procedure: BeckupEdgeNode
1: ***for*** *(content x_i in Node n_i)*
2: *compare conetnt x_i with content x_j*
3: ***if*** *(R.content x_j > R.content x_i)* ***then***
4: *least downloaded content = content x_i*
5: *evict content x_i from Node n_i*
6: cache *content* x_i at Node n_k
7: ***end if***
8: ***end for***
9: ***return*** content x_i cached at node n_k
[where $i, j = \{1, 2, 3, \dots \dots n\}$ and $i \neq j \neq k, k = n - 1$, R = requests]
[where n_i= intermediate node and n_k= edge node, $x_i \in X, x_j \in X, n_k \in X$]

Algorithm 3 shows the content caching mechanism. In backup facility procedure, when a high prioritized content is arrived at intermediate node, the Time Aware Least Downloaded (TALD) content is evicted from the intermediate node to accommodate the new high prioritized content and the evicted content in cached at backup facility node. The backup facility node is considered as edge node because there is less probability to generate a large number of subsequent requests for TALD contents. Therefore, the PCPA manages the cache by accommodating the high prioritized content at intermediate nodes. Thus, the TALD content is cached at edges of the NDN-based IoT network. Algorithm 4 shows the mechanism of backup caching method. Moreover, the cache of the intermediate nodes will be used for the high prioritized content efficiently. As a result, the subsequent requests for TALD content can accomplish at one-hop distance and hence, the cache hit will be maximized with less response latency. To understand the PCPA concepts, Figure 2 illustrates the content selection, content caching, and backup caching mechanism of the PCPA caching strategy. In the given scenario, four requests were sent from IoT-based devices (1, 3, and 4) to fetch content C1. Instantly, another request is sent by IoT-based device 5 to download content C1. According to the PCPA, the content that has received a higher number of user requests and all the requests have traveled more distance in hops will be selected as popular. Therefore, all the requests for content C1 have traveled more hops to get the content. Thus, content C1 is selected as popular and recommended to cache along the data downloading nodes. In order to cache C1 along the path, the number of incoming users' requests is measured at each node.

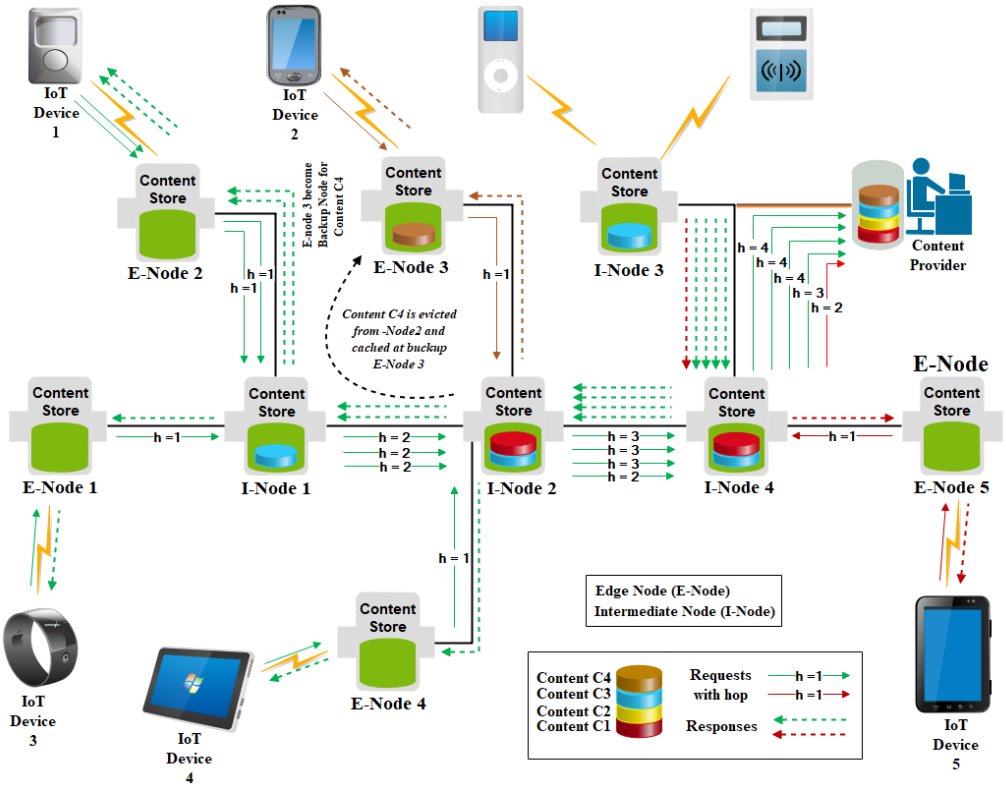

FIGURE 2. PRIORITY BASED POPULARY-AWARE CACHING

If a node shows that the number of requests for content C1 is more than threshold 2, the content C1 will be cached at that node. Consequently, C1 is cached at intermediate nodes 4 and 2 because these nodes are showing more requests than the threshold requests of the total number of users' requests generated at the content provider. As content C1 needs to be cached at I-Node 2, the cache of that node is already occupied by highly requested content C1 and C2. According to PCPA, content C4 is evicted based on the time-aware least downloaded content from I-Node 2 and cached at the edge node, E-Node 3, to make room for incoming content C1 at I-Node 2.

V. PERFORMANCE EVALUATION

To evaluate the performance of the proposed PCPA caching strategy and the benchmark strategies, the network simulator NDNSim version has been selected. For the present research, a simulation environment has been established to evaluate the proposed NDN-based IoT caching strategy. The requirements and platform for the simulation are outlined below. The simulation setup has been constructed to measure the proposed model's performance and validation. Using a network simulator to study NDN-IoT caching performance offers several advantages for this research. NDNSim simulator provides a set of application helper classes and references to measure several aspects of NDN-based IoT strategies under different scenarios. Different researchers choose different metrics according to their research area and the proposed model. Therefore, for this research, some metrics have been selected to test the effectiveness of the proposed content deployment strategy. The metrics used to evaluate the performance of PCPA are content retrieval latency, cache hit ratio, average hop count, and total energy consumption. The PCPA has been simulated using a desktop machine with the following specifications: an i7 computer core with 12 GB RAM, a microprocessor of 3 GHz processor, and Ubuntu 20.04 version installed as an operating system to run the NDNSim simulator. For the current simulation, 200 IoT nodes have been randomly distributed in a square area of 500m x 500m grid topology.

Fifteen nodes at the border were selected as content producers (publishers), 45 nodes were selected as cache nodes, and 140 nodes were chosen as users (subscribers). The Zipfian distribution was used to generate contents, and its α distribution value ranged between 0.2 to 1.0. IEEE802.11a was selected as the wireless interface (wireless technology), and the cache size of each node varied from 5 chunks to 20 chunks. The simulation was executed 50 times, and the percentage of these was considered the final result. The parameters and selected values for the present simulation are given in the following Table 1. In simulation, the cache size was selected as constant (minimum and

maximum) such as 5 chunks and 20 chunks. While, the weight factor (α) was selected as varying parameter and its value is taken as α = 0.2 to α = 1.0.

TABLE 1. Parameters and Corresponding Values

Parameters	Values
Wireless Technology	IEEE 802.11
Topology Size	(500 × 500) m
Radio Coverage Area	100m
Total number of nodes	200
Size of a chunk	200 bytes
Cache Size	5, 10, 15, and 20 Chunks
Number of Consumers	140 nodes
Mobility model	Random
Popularity model	Zipf
Weight Factor (α)	0.2, 0.4, 0.6, 0.8, 1.0
Simulation Time	300s
Number of simulations Run	50 runs

A. COMPARING STRATEGIES

Four caching strategies were selected as Cache Everything Everywhere (CEE), Periodic Caching Strategy (PCS), Endpoint Linked Green Content (ELGC) caching strategy, and Energy-aware Caching Placement (ECP) strategy to compare the performance of the proposed PCPA caching strategy with related caching strategies in the NDN-based IoT environment.

1. Cache Everything Everywhere

Cache Everything Everywhere (CEE) [45] is known as NDN-IoT default caching strategy. It provides a simple structure that offers high data availability and reachability.

2. Periodic Caching Strategy

Periodic Caching Strategy (PCS) [31] was proposed to provide caching of the popular contents at the betweenness centrality node and when it gets filled the coming contents are cached at the edge node.

3. Energy-aware Caching Placement

Energy-aware Caching Placement (ECP) [46] strategy was proposed to improve the energy efficiency of the network. Its goal is to optimize the energy by trading off between the content caching energy and content transmission energy.

4. Endpoint Linked Green Content

The Endpoint Linked Green Content (ELGC) [18] caching strategy recently been proposed to increase the energy efficiency using threshold-based data offloading mechanism. It optimizes the load sharing between rim nodes and end-devices for the caching of the incoming contents.

B. EVALUATION METRICS

Based on the proposed model, the simulation was run on the Cache Hit Ratio (CHR), Average Hop Count (AHC), Content Retrieval Latency (CRL), and Energy Consumption (EC).

1 Cache Hit Ratio

Cache Hit Ratio (CHR) is referring to the total number of satisfied user incoming requests from the network caching nodes. It is ratio between hit rate and miss rate. CHR is measured by combining the successful data transmissions sent to the requested users and unsatisfied user requests.

2. Average Hop Count

Average Hop Count (AHC) denotes that the total number of hops is required for a request to travel from the user to the source caching node where the cache hit occurs. When a user sends out some requests to fetch some content, the requests travels through the network caching nodes hop by hop till the hit occur. Therefore, all the hops are measured for all requests that sent to fetch same content is known as AHC. The AHC is shorter for the content fetched from the caching node as compared to the data found from the publisher.

3. Content Retrieval Latency

Content Retrieval Latency (CRL) refers to the time unit in millisecond that is used to consume within the multiple links taken to reply to a coming request. It can be defined as the time required to complete one transmission of a user request that is sent by the user to fetch a desirable content or data item from the caching node. Usually, in NDN-IoT caching environment, CRL is measured for the whole network in the form of average latency and it can be defined as the time is required in the transmission of the overall users' requests to fetch the demanded contents.

3. Energy Consumption

Energy Consumption (EC) is referred to the total energy consumed by caching nodes in NDN-based IoT network and it is measured the total of the whole network to evaluate the performance. The IoT-based devices have limited powered batteries that make the energy performance significant to enhance the overall system performance. The energy consumption is considered as the most important parameter in developing a new caching strategy for NDN-based IoT scenarios because of energy constrained IoT devices. Therefore, a caching strategy is considered as efficient if, it has the ability to reduce the energy consumption by caching the popular contents at the nodes that can to fulfill the future demands of the end users. Thus, the caching of disseminated content has different affect over the battery-life of the IoT devices such as the data closer, a few nodes will be used to transmit the content to the end users and the less energy will be consumed as compared to the content fetched far from the end users because content needs to travel a greater number of nodes. For the energy consumption additional parameters are selected such as the initial energy is taken as 1 jule, energy required for a transaction is 50 nJ/bit, and energy for caching content is 10 nJ/bit.

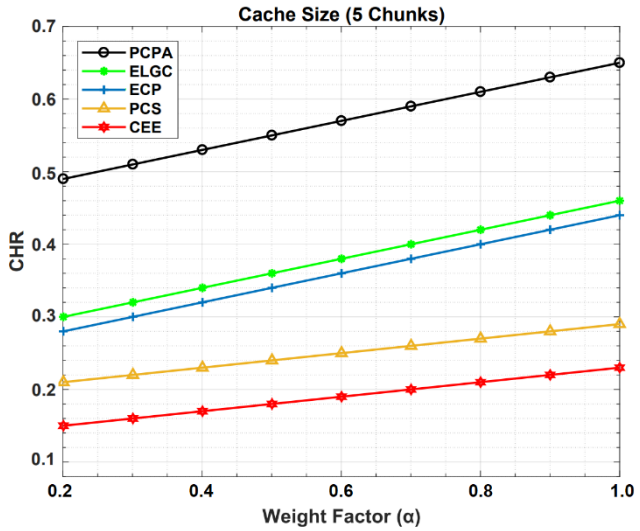

FIGURE 3. CACHE HIT RTAIO ON CACHE SIZE (5 CHUNKS)

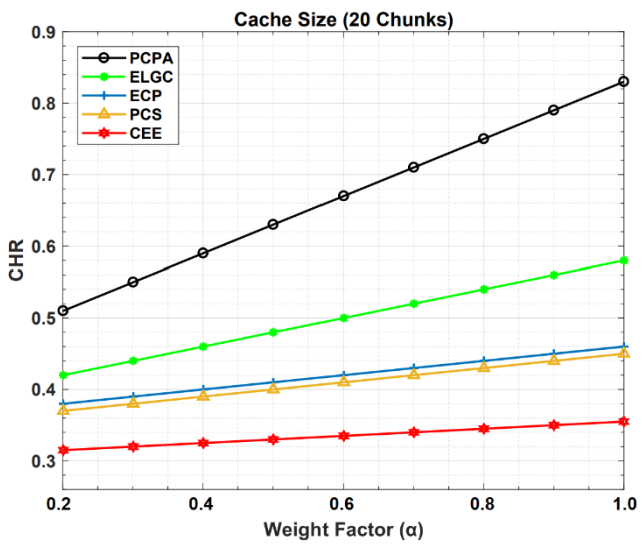

FIGURE 4. CACHE HIT RTAIO ON CACHE SIZE (20 CHUNKS)

B. RESULTS AND DISCUSSION

Different caching strategies were selected to compare the performance of the proposed PCPA caching strategy with related caching strategies in the NDN-based IoT environment. The CHR performance of PCPA was compared with CEE, PCS, ECP, and ELGC to check the significance of the outcomes. Figure 3 and Figure 4 shows the simulation results on CHR. It is clear from Figure 3 and Figure 4 that the performance improves with increasing the value of the Weight Factor (α).

In this experiment, the weight factor is taken varying from 0.2 to 1.0 with constant cache size that is selected as 5 chunks and 20 chunks. However, the performance on EC appearances is different as compared to the CHR, CRL, and AHC. The reason is that, in EC, the consumption is increased with higher content popularity value and with large cache size because with higher popularity value more users request is generated to fetch the high popular content and a large number of contents are accommodated in large cache size. Therefore, the energy consumption is increased in accommodating a large number of contents with higher popularity value as 1.0.

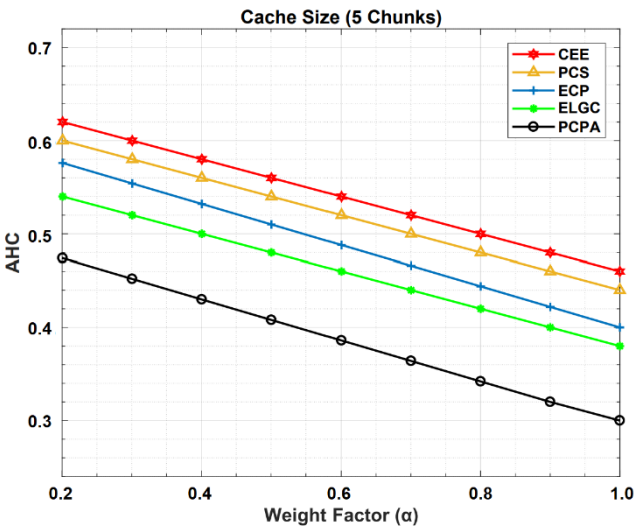

FIGURE 5. AVERAGE HOP COUNT ON CACHE SIZE (5 CHUNKS)

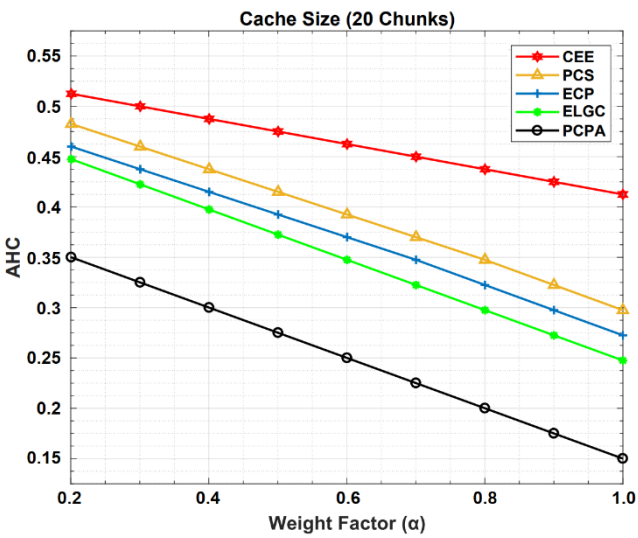

FIGURE 6. AVERAGE HOP COUNT ON CACHE SIZE (20 CHUNKS)

The PCPA achieves better CHR performance throughout the simulation process with each constant cache size and varying α values. CEE performs poorly and achieves the lowest result with higher weight factor at α = 1.0. While. PCS and ECP performs better than CEE with a lower α value and higher α value as α =0.2 and α = 1.0. Moreover, ELGC outperformed than PCS and ECP because it implements the function of caching contents at the end devices that increases the cache hit for the subsequent requests. Therefore, ELGC performs better than ECP, PCS and achieves the higher result as 0.587

at α = 1.0. However, the proposed PCPA strategy beats all the comparing strategies and achieves the highest result at α = 1.0, as shown in Figure 3 and Figure 4.

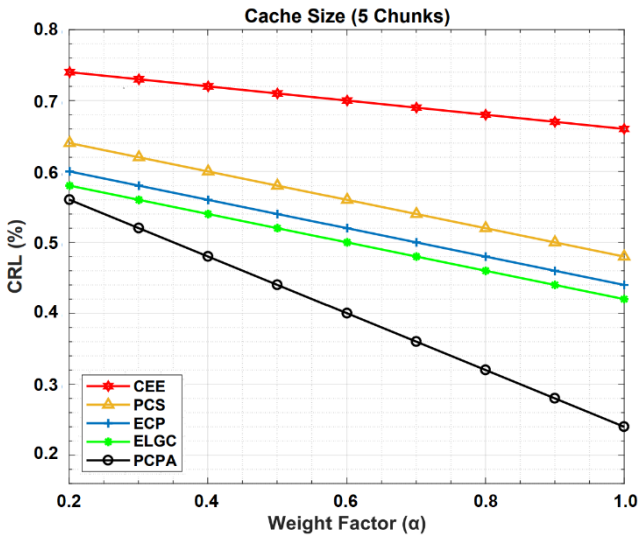

FIGURE 7. CONTENT RETRIEVAL LATENCY ON CACHE SIZE (5 CHUNKS)

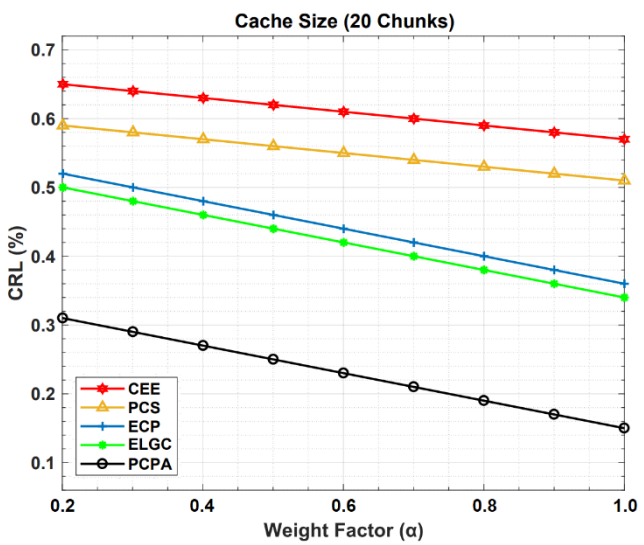

FIGURE 8. CONTENT RETRIEVAL LATENCY ON CACHE SIZE (20 CHUNKS)

Figure 5 and Figure 6 show the impact of a constant cache size with varying weight factor (α) on the Average Hop Count (AHC). The PCPA achieves better AHC performance throughout the simulation process for each constant cache size and varying α values. CEE performs poorly and achieves the lowest result throughout the varying α parameter from α = 0.2 to α = 1.0. PCS performs better than CEE with a lower α value and achieves the better outcomes regarding AHC result at α = 1.0. However. ECP performs slightly better than PCS with both cache size (5 chunks and 20 chunks). Moreover, ELGC performs better than CEE, PCS, and ECP by achieving the better AHC result with as α parameters. However, the proposed PCPA strategy beats all the comparing strategies and showing better results with both cache sizes and α parameters.

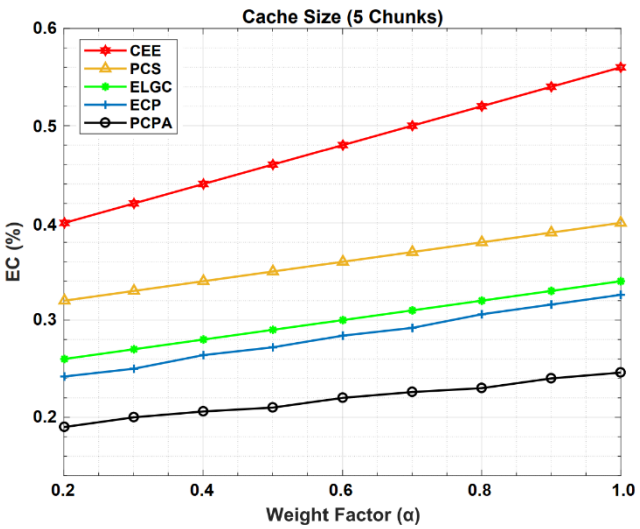

FIGURE 9. ENERGY CONSUMPTION ON CACHE SIZE (5 CHUNKS)

Figure 7 and Figure 8 illustrates the simulation results on CRL using two different constant cache sizes, 5 chunks, and 20 chunks. The x-axis shows the varying weight factor α, while the y-axis shows CRL performance in average or percentage. CEE performs poorly in terms of CRL, while PCS performs better than CEE. Moreover, ECP and ELGC perform relatively similarly and shows better performance than CEE and PCS with all α values. However, the PCPA outperforms all strategies throughout the simulation process with all α values and achieves better results than CEE, PCS, ECP and ELCG. PCPA outperforms because of its ability to cache popular and less popular content near end-users, thereby achieving better performance. The EC performance shows different results as compared to the CHR, CRL, and AHC. The reason is that, in EC, the consumption is increased with higher content popularity value and with large cache size because with higher popularity value more users request is generated to fetch the high popular content and a large number of contents are accommodated in large cache size. Therefore, the energy consumption is increased in accommodating a large number of contents with higher popularity value. Figure 9 and Figure 10 shows the effect of constant cache size on different simulation scenarios using varying weight factors. As the α value increases the energy consumption is increased in both simulation scenarios because, with higher popularity values a large number of requests are received and the corresponding content are sent back to the end users. Therefore, the network usage is increased and consequently, the energy consumption gets increased. However, the proposed PCPA caching strategy outperform with both caching sizes than CEE, PCS, ECP, and ELGC. The PCPA caches popular content close to the end users and less energy is consumed in fetching the

demanded contents and thus overall energy consumption is reduced.

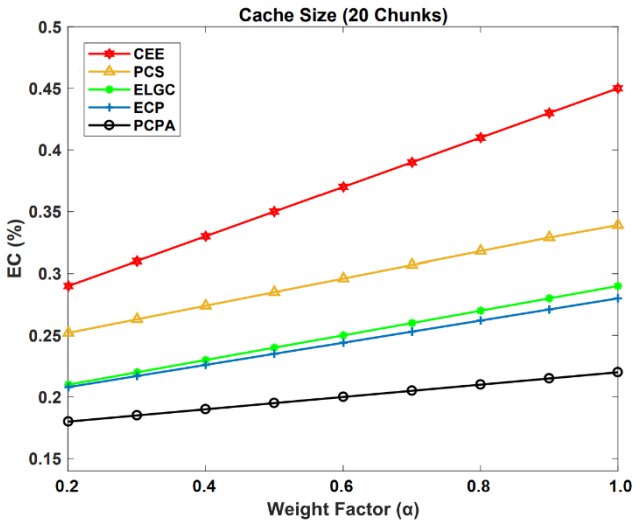

FIGURE 10. ENERGY CONSUMPTION ON CACHE SIZE (20 CHUNKS)

On the other hand, the CEE perform poor and increases the energy consumption during data dissemination. The reason is that, CEE performs caching operations at each node along the data routing path and consequently, the content evictions is also increased that increases the usage of the network and hence, the overall energy consumption is maximized as shown in Figure 9 and Figure 10. The ELCG and ECP performed relatively with both cache sizes due to their caching structures of caching contents. However, the energy consumption is higher in ELGC than ECP because ELGC executes extra function during content caching and content replacement and consumes additional energy during offloading the contents. ECP performs somehow better than ELCG because ECP computes less parameters as compare to the ELGC for making a caching decision. Therefore, ECP consumes less energy than ELCG and improve the overall performance in terms of energy consumption.

VI. CONCLUSION AND FUTURE WORK

NDN-based IoT networks offer the potential to create an Internet environment through caching, providing various benefits such as decoupling receivers from producers, reducing redundant data dissemination, and enhancing scalability. However, despite these advantages, NDN-based IoT caching faces several challenges. This study aims to review different caching concepts and issues in NDN-IoT caching environments. The importance of NDN and IoT architectures, as well as caching concepts, are introduced. Existing caching strategies are identified, and the problems associated with them are discussed. To address these issues, a new caching strategy called Priority-based Content Popularity-Aware (PCPA) Caching is proposed. The PCPA strategy is implemented using the NDNSim simulator, and its performance is compared with other caching strategies such as CEE, PCS, ELCG, and ECP. Metrics such as latency, stretch ratio, and cache hit ratio are used to evaluate the caching performance, and the results show that the PCPA caching strategy outperforms the other strategies, especially with a large cache size.

In NDN, the caching module plays a crucial role in improving network performance. The placement and replacement of cached content are vital tasks for efficient caching. NDN caching strategies are particularly suitable for enhancing IoT-based network performance. It is important to select a suitable caching node in the network that can provide efficient caching performance in terms of data availability, low latency, and reduced energy consumption. However, content popularity can have a varying impact on caching performance. Highly popular content that is rarely used in the future can lead to inefficient energy utilization and excessive storage consumption. Frequently requested content that cannot fit into a node's cache may result in a large number of requests being forwarded to remote providers, significantly increasing content retrieval delays. While NDN-IoT caching offers numerous benefits, it also imposes complexities and restrictions on caching strategy development in different IoT networks. Therefore, the proposed caching strategy considers content-based properties such as freshness, content popularity, and provider, as well as node properties including path stretch, battery power level, and free memory, to enhance overall caching performance.

Based on these properties, the proposed caching strategy can be applied to various Internet technologies such as Software Defined Networking (SDN), Blockchain, cloud computing, fog computing, edge computing, and mobile edge computing. Furthermore, the proposed caching model can be utilized to evaluate and enhance network performance in terms of bandwidth consumption, memory consumption, and backhaul consumption. Additionally, the proposed strategy can be implemented to improve caching performance in Ad hoc Networks such as Mobile Ad hoc Networks (MANET) and Vehicular Ad hoc Networks (VANET). In conclusion, the proposed strategy holds potential for improving caching performance in 5G/6G cellular networks.

REFERENCES

[1] D. R. d. S. Medeiros and M. A. Fernandes, "Distributed genetic algorithms for low-power, low-cost and small-sized memory devices," *Electronics,* vol. 9, p. 1891, 2020.

[2] P. Perazzo, F. Righetti, M. La Manna, and C. Vallati, "Performance evaluation of attribute-based encryption on constrained iot devices," *Computer Communications,* vol. 170, pp. 151-163, 2021.

[3] D. D. F. Del Rio, B. K. Sovacool, N. Bergman, and K. E. Makuch, "Critically reviewing smart home technology applications and business models in Europe," *Energy Policy,* vol. 144, p. 111631, 2020.

[4] N. A. Khan and A. Awang, "Elliptic Curve Cryptography for the Security of Insecure Internet of Things," in *2022 International Conference on Future*

Trends in Smart Communities (ICFTSC), 2022, pp. 59-64.

[5] H. Khalajzadeh, A. J. Simmons, M. Abdelrazek, J. Grundy, J. Hosking, and Q. He, "An end-to-end model-based approach to support big data analytics development," *Journal of Computer Languages*, vol. 58, p. 100964, 2020.

[6] F. Firouzi, B. Farahani, M. Weinberger, G. DePace, and F. S. Aliee, "Iot fundamentals: Definitions, architectures, challenges, and promises," *Intelligent Internet of Things: From Device to Fog and Cloud*, pp. 3-50, 2020.

[7] G. Gardašević, M. Veletić, N. Maletić, D. Vasiljević, I. Radusinović, S. Tomović, and M. Radonjić, "The IoT architectural framework, design issues and application domains," *Wireless personal communications*, vol. 92, pp. 127-148, 2017.

[8] X. Wang and X. Qian, "Toward Named Data Networking: An Approach Based the Internet of Things Cloud With Edge Assistance," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 8, pp. 21-27, 2022.

[9] C. Gündogan, P. Kietzmann, T. C. Schmidt, M. Lenders, H. Petersen, M. Wählisch, M. Frey, and F. Shzu-Juraschek, "Information-centric networking for the industrial IoT," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, 2017, pp. 214-215.

[10] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent advances in information-centric networking-based Internet of Things (ICN-IoT)," *IEEE Internet of Things Journal*, vol. 6, pp. 2128-2158, 2018.

[11] D. Mars, S. Mettali Gammar, A. Lahmadi, and L. Azouz Saidane, "Using information centric networking in internet of things: a survey," *Wireless personal communications*, vol. 105, pp. 87-103, 2019.

[12] A. Djama, B. Djamaa, and M. R. Senouci, "Information-Centric Networking solutions for the Internet of Things: A systematic mapping review," *Computer Communications*, vol. 159, pp. 37-59, 2020.

[13] W. Fang, M. Xu, C. Zhu, W. Han, W. Zhang, and J. J. Rodrigues, "FETMS: Fast and efficient trust management scheme for information-centric networking in Internet of Things," *IEEE access*, vol. 7, pp. 13476-13485, 2019.

[14] J. Li, B. Liu, and H. Wu, "Energy-efficient in-network caching for content-centric networking," *IEEE Communications Letters*, vol. 17, pp. 797-800, 2013.

[15] J. Pfender, A. Valera, and W. K. Seah, "Performance comparison of caching strategies for information-centric IoT," in *Proceedings of the 5th ACM conference on information-centric networking*, 2018, pp. 43-53.

[16] N. A. Khan, A. Awang, and S. A. B. A. Karim, "Security in Internet of Things: A review," *IEEE access*, 2022.

[17] N. Askar, A. Habbal, F. Z. Alden, X. Wei, H. Alaidaros, J. Guo, and H. Yu, "Forwarding Strategies for Named Data Networking based IOT: Requirements, Taxonomy, and Open Research Challenges," *IEEE access*, 2023.

[18] H. Shrisha and U. Boregowda, "An energy efficient and scalable endpoint linked green content caching for Named Data Network based Internet of Things," *Results in Engineering*, vol. 13, p. 100345, 2022.

[19] K. K. Singh and R. K. Dudeja, "Hybrid Information Placement in Named Data Networking-Internet of Things System," in *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2022, pp. 593-601.

[20] M. S. M. Shah, Y.-B. Leau, Z. Yan, and M. Anbar, "Hierarchical naming scheme in named data networking for Internet of Things: A review and future security challenges," *IEEE access*, vol. 10, pp. 19958-19970, 2022.

[21] M. A. Naeem, T. N. Nguyen, R. Ali, K. Cengiz, Y. Meng, and T. Khurshaid, "Hybrid cache management in IoT-based named data networking," *IEEE Internet of Things Journal*, vol. 9, pp. 7140-7150, 2021.

[22] S. Hussain, S. S. Ullah, A. Gumaei, M. Al-Rakhami, I. Ahmad, and S. M. Arif, "A novel efficient certificateless signature scheme for the prevention of content poisoning attack in named data networking-based internet of things," *IEEE access*, vol. 9, pp. 40198-40215, 2021.

[23] B. Alahmri, S. Al-Ahmadi, and A. Belghith, "Efficient pooling and collaborative cache management for NDN/IoT networks," *IEEE access*, vol. 9, pp. 43228-43240, 2021.

[24] W. M. H. Azamuddin, A. H. M. Aman, R. Hassan, and T.-A. N. Abdali, "Named data networking mobility: A survey," in *Emerging Technology Trends in Internet of Things and Computing: First International Conference, TIOTC 2021, Erbil, Iraq, June 6–8, 2021, Revised Selected Papers*, 2022, pp. 266-281.

[25] M. A. Naeem, S. A. Nor, S. Hassan, and B.-S. Kim, "Performances of probabilistic caching strategies in content centric networking," *IEEE access*, vol. 6, pp. 58807-58825, 2018.

[26] M. A. Naeem, M. A. U. Rehman, R. Ullah, and B.-S. Kim, "A comparative performance analysis of popularity-based caching strategies in named data networking," *IEEE access*, vol. 8, pp. 50057-50077, 2020.

[27] M. Amadeo, "A literature review on caching transient contents in vehicular named data networking," in *Telecom*, 2021, pp. 75-92.

[28] S. Wang, H. Chen, and Y. Wang, "Collaborative caching for energy optimization in content-centric internet of things," *IEEE Transactions on Computational Social Systems,* vol. 9, pp. 230-238, 2021.

[29] H. Al-Ward, C. K. Tan, and W. H. Lim, "Caching transient data in Information-Centric Internet-of-Things (IC-IoT) networks: A survey," *Journal of Network and Computer Applications,* p. 103491, 2022.

[30] M. Amadeo, C. Campolo, G. Ruggeri, and A. Molinaro, "Beyond edge caching: Freshness and popularity aware iot data caching via ndn at internet-scale," *IEEE Transactions on Green Communications and Networking,* vol. 6, pp. 352-364, 2021.

[31] M. A. Naeem, R. Ali, B.-S. Kim, S. A. Nor, and S. Hassan, "A periodic caching strategy solution for the smart city in information-centric Internet of Things," *Sustainability,* vol. 10, p. 2576, 2018.

[32] J. Pfender, A. Valera, and W. K. Seah, "Reassessing caching performance in information-centric IoT," *Internet of Things,* vol. 18, p. 100479, 2022.

[33] M. A. Naeem, S. A. Nor, S. Hassan, and B.-S. Kim, "Compound popular content caching strategy in named data networking," *Electronics,* vol. 8, p. 771, 2019.

[34] M. A. Naeem, R. Ullah, Y. Meng, R. Ali, and B. A. Lodhi, "Caching content on the network layer: a performance analysis of caching schemes in ICN-based Internet of Things," *IEEE Internet of Things Journal,* vol. 9, pp. 6477-6495, 2021.

[35] Y. Meng, M. A. Naeem, R. Ali, and B.-S. Kim, "EHCP: An efficient hybrid content placement strategy in named data network caching," *IEEE access,* vol. 7, pp. 155601-155611, 2019.

[36] Y. Meng, M. A. Naeem, M. Sohail, A. K. Bashir, R. Ali, and Y. B. Zikria, "Elastic caching solutions for content dissemination services of ip-based internet technologies prospective," *Multimedia Tools and Applications,* vol. 80, pp. 16997-17022, 2021.

[37] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "Caching in named data networking for the wireless internet of things," in *2015 international conference on recent advances in internet of things (RIoT)*, 2015, pp. 1-6.

[38] C. Bernardini, T. Silverston, and O. Festor, "MPC: Popularity-based caching strategy for content centric networks," in *2013 IEEE international conference on communications (ICC)*, 2013, pp. 3619-3623.

[39] J. Ren, W. Qi, C. Westphal, J. Wang, K. Lu, S. Liu, and S. Wang, "Magic: A distributed max-gain in-network caching strategy in information-centric networks," in *2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, 2014, pp. 470-475.

[40] A. M. Alberti, M. A. F. Casaroli, D. Singh, and R. da Rosa Righi, "Naming and name resolution in the future internet: Introducing the NovaGenesis approach," *Future Generation Computer Systems,* vol. 67, pp. 163-179, 2017.

[41] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. L. Aguiar, and A. V. Vasilakos, "Information-centric networking for the internet of things: challenges and opportunities," *IEEE Network,* vol. 30, pp. 92-100, 2016.

[42] M. Amadeo, G. Ruggeri, C. Campolo, A. Molinaro, and G. Mangiullo, "Caching popular and fresh IoT contents at the edge via named data networking," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 610-615.

[43] S. Alduayji, A. Belghith, A. Gazdar, and S. Al-Ahmadi, "PF-ClusterCache: Popularity and Freshness-Aware Collaborative Cache Clustering for Named Data Networking of Things," *Applied Sciences,* vol. 12, p. 6706, 2022.

[44] S. Vural, N. Wang, P. Navaratnam, and R. Tafazolli, "Caching transient data in internet content routers," *IEEE/ACM Transactions on Networking,* vol. 25, pp. 1048-1061, 2016.

[45] D. Gupta, S. Rani, S. H. Ahmed, and R. Hussain, "Caching policies in NDN-IoT architecture," *Integration of WSN and IoT for Smart Cities,* pp. 43-64, 2020.

[46] O. Serhane, K. Yahyaoui, B. Nour, and H. Moungla, "Energy-aware cache placement scheme for iot-based icn networks," in *ICC 2021-IEEE International Conference on Communications*, 2021, pp. 1-6.

Yahui Meng received the B.S. degree in computer science and technology from Air Force Engineering University, Xi'an, China, in 2003 and the M.S degree in software engineering from Huazhong University of Science and Technology, Wuhan, China, in 2008. He is currently pursuing the Ph.D. degree in computer science, he is a Ph.D. Scholar at the InterNetWorks Research Lab, School of Computing, Universiti Utara Malaysia. Meanwhile, he is

also currently an associate professor in School of Science, Guangdong University of Petrochemical Technology, Maoming, China. His major interests are in the field of information centric wireless networks, named data networking and the Internet of Things.

Amran bin Ahmad: Dr. Amran Ahmad is a Senior Lecturer in the School of Computing and a member of the Internetworks Research Lab (IRL) at Universiti Utara Malaysia. He received his Ph.D. in Network Security from the same university. As a member of IRL, he is actively involved in all activities initiated by the lab, including being an IPv6 trainer, a committee member of the International Conference on Internet Applications, Protocols, and Services (Netapps). Dr. Amran is also a member of the Internet Society professional body and contributes to them as an IRL member. His research interests include Mobile Network Technology, Web Services, and E-Learning. He has conducted several research projects supported by University and FRGS grants in the area of Mobile Network Technology and E-Learning. Dr. Amran has published several peer-reviewed articles in leading journals and conferences. He has also served as a program committee member for several conferences in the field of Mobile Network Technology and Web Services. His research work has been presented at numerous international and local conferences. In addition to his research work, Dr. Amran is passionate about teaching and mentoring. He has developed and taught several undergraduate and graduate courses in Computer Science, with a focus on Mobile Network Technology, Web Services, and E-Learning. He has supervised numerous research projects and thesis work related to these areas. Dr. Amran is also actively involved in promoting diversity and inclusion in STEM fields.