Received 3 April 2024, accepted 28 May 2024, date of publication 3 June 2024, date of current version 30 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3409209

RESEARCH ARTICLE

Enhancement of Fog Caching Using Nature Inspiration Optimization Technique Based on Cloud Computing

MOHAMED R. ELNAGAR[1], AHMED AWAD MOHAMED[2], BENBELLA S. TAWFIK[1], AND HOSAM E. REFAAT[1]

[1]Information System Department, Faculty of Computers and Information, Suez Canal University, Ismailia 41522, Egypt

[2]Information System Department, Cairo Higher Institute for Languages and Simultaneous Interpretation, and Administrative Science, Cairo 11571, Egypt

Corresponding author: Mohamed R. Elnagar (mohamedelnagar6102603@gmail.com)

ABSTRACT Caching plays an important role in reducing latency and increasing the overall performance of fog computing systems. With the rise of the Internet of Things (IoT) and edge computing, fog computing has become an essential paradigm for addressing the challenges related to latency-sensitive and bandwidth-intensive applications. The integration of Information-Centric Networking (ICN) and Fog Computing (ICN-Fog) has emerged as a promising solution for meeting the demands of low-latency and high-throughput applications in rapidly growing IoT devices. A step was taken by ICN-Fog to reduce latency and achieve higher data communication and information gathering for fog computing. Using Artificial Intelligence (AI) methods, the Firefly Optimization Technique was introduced as an effective optimization algorithm to enhance the caching technique. In this study, we aim to improve several performance metrics, such as the cache hit ratio, internal link load, and average query duration. To achieve these enhancements, we propose a unique solution inspired by the Firefly Optimization Technique. This technique applies to the ICN-Fog caching model, which was utilized to intelligently determine cache placement and network topology adaptations. CloudSim was employed to execute a simulation of the proposed strategy. The results of the experiments suggest that the Multi-Objective Firefly Algorithm (MOFA) outperforms the compared algorithms in terms of both efficiency and effectiveness in identifying the optimal caching technique.

INDEX TERMS Cloud, fog computing, information centric network (ICN), ICN-fog, caching, multi-objective optimization, firefly algorithm and CloudSim.

I. INTRODUCTION

Cloud Computing (CC) allows easy and efficient access, storage, and management of data and applications over the Internet without the need for local hardware or infrastructure. It offers a range of services tailored to meet individual and enterprise needs. Cloud services have three primary models: Infrastructure as a Service (IaaS), which allows users to use IT infrastructures like processing, storage, and networking resources; Platform as a Service (PaaS), which lets users create software and provides support for the software development lifecycle, often using middleware for software management and configuration; and Software as a Service (SaaS), which delivers software applications through the cloud on a subscription basis and offers on-demand access to these applications [1]. However, Cloud computing offers benefits, such as scalability, cost-effectiveness, accessibility, reliability, and innovation. There are also limitations and obstacles, including data storage considerations, latency, and costs related to data transfer and bandwidth usage. So, The Cloud can be managed more efficiently using alternative paradigms such as fog computing and information-centric networking (ICN). Fog computing distributes computing resources closer to the edge, whereas ICN focuses on

The associate editor coordinating the review of this manuscript and approving it for publication was Paulo Mendes.

transmitting content to end users. Together, they can create efficient cloud computing environments. In addition, an ICN can serve as the underlying network architecture for fog computing, providing a better way to access and distribute data in fog computing environments. Fog is a computing and networking framework that aims to reshape the IoT, 5G, and embedded AI applications [2]. Rather than relying solely on the cloud for support, fog facilitates storage, communication, and networking directly at or near the end user [3]. Information-Centric Networking (ICN) is a new networking approach focusing on content delivery. It eliminates the need for direct communication between the source and destination and precise location knowledge. Instead, it emphasizes identifying and transmitting specific named content. Unlike current IP networks, ICN has an intelligent data plane that facilitates adaptable and effective data dissemination within the network layer [4]. ICN faces various challenges, including naming [5], [6], content discovery [7], and caching [8]. ICN-Fog is an architecture that combines Information-Centric Networking and Fog Computing. It introduces a ''Fog-to-Fog'' layer adhering to ICN principles to enhance content delivery and caching in edge environments. This promotes efficient data distribution and reduces reliance on traditional communication models.

A. MOTIVATION

When it comes to data-heavy tasks such as video streaming and IoT, traditional cloud-based methods often experience latency issues. However, fog caching provides a solution by strategically placing cached content on fog nodes located at the edge of the network. This approach reduces data traversal distances, thereby improving the overall user experience and aligning with the growth of IoT. Additionally, fog caching makes use of previously underutilized fog resources to minimize congestion and bandwidth costs. This concept is similar to the light of a firefly, guiding us towards a more efficient and interconnected digital infrastructure. In the same way that a firefly illuminates the night, fog caching illuminates the path towards a more responsive and interconnected network. Instead of data being confined to distant centers, fog caching allows generating nodes to embrace it. Ultimately, fog caching creates a streamlined, adaptable, and interconnected digital infrastructure, much like the fleeting brilliance of a firefly's light.

B. CONTRIBUTIONS

In this study, a new and improved version of fog caching is introduced, which utilizes a Multi-Objective Firefly Algorithm (MOFA). This updated system is designed to optimize the placement of cached content in fog nodes, with the aim of maximizing the cache hit ratio, reducing latency, and minimizing internal link load. The Firefly Algorithm (FA) is used to distribute cache content based on user access patterns and to adapt to changing demands, resulting in a more adaptive approach than traditional caching methods. As a result, cache hit ratios are significantly improved, leading to reduced content retrieval latencies for end-users. The integration of MOFA with fog caching represents an innovative advancement in optimizing fog-computing performance, as MOFA excels in solving multi-objective problems by considering conflicting objectives and achieving trade-offs between the cache hit ratio, latency reduction, and internal link load. This approach enhances cache hit ratio, reduces latency by strategically placing caches, and minimizes the internal link load, resulting in a more responsive system.

C. PAPER ORGANIZATION

The structure of this paper is as follows. Section II provides a literature review of the relevant prior research. In Section III, we explore the information-centric network and its connection to the Internet of Things (IoT) and fog layers. Section IV introduces the concept of Caching, and Section V explains the firefly algorithm. Section VI demonstrates the proposed model (MOFA) and Section VII details the system implementation. Section VIII outlines the experimental setup; Section IX specifies the simulation parameters and Section X explains the performance metrics. In Section XI, we discuss and analyze the results. Finally, Section XII concludes the study and highlights future research areas.

II. RELATED WORK

Many studies in the literature have addressed fog caching techniques in a cloud environment, as follow:

Yining Huaa et al. Introduced a fog-caching system for IoT applications. Their approach had three key features: utilizing in-network resources and end-user devices, strategically placing caching nodes along the delivery path, and integrating reactive and proactive techniques. Their simulations in the Icarus environment outperformed the eight established benchmarks, showing significant improvements in the internal link load, path stretch, and cache hit ratio. The proposed scheme also has competitive latency performance, especially in densely concentrated content-distribution scenarios [9].

Muhammad Ali Naeem et al. Explored fog computing, Internet of Things (IoT), and challenges surrounding data security and distribution within the realm of fog computing. They also delved into fog-based caching strategies that aim to address common challenges in fog computing. Their work introduced multiple caching techniques and explained their contributions and advantages as well as how they help overcome obstacles in fog computing. In addition, they discussed the integration of machine learning methodologies to enhance cache security and administration in fog-computing environments. Finally, they outlined several promising areas for future research in the domains of caching, fog computing, and machine learning within the context of fog computing [10].

Ibrahim Abdullahi et al. introduced a new method of integrating Information-Centric Networking (ICN) into the world of ubiquitous computing. The approach involves using an

ICN as an API and focusing on implementing fog computing techniques to establish caches at edge nodes. This method uses names to reference objects instead of traditional IP addresses, which significantly improves the proximity of cloud-based information on the Internet of Everything to end-users. Fog computing is utilized to deploy edge-processing nodes and facilitate caching, allowing the ICN to function off the conventional network path. The fog concept accommodates heterogeneity within the Internet of Things (IoT) by representing edge nodes responsible for computing, processing, and storage [11].

Dinh Nguyen et al. Introduced ICN-Fog, a groundbreaking horizontal layer that enables fog-to-fog communication using Information-Centric Networking (ICN). By implementing the ICN-Fog, the Fog layer can transmit data horizontally, process data across multiple fog nodes in a decentralized manner and offer seamless mobility assistance through intelligent connectionless name-based communication between fog entities. Their study explained the fundamental principles that guided their design approach and demonstrated the advantages of this innovative fog architecture using two case studies [12].

Maroua Meddeb et al. Explored the difficulties related to caching in IoT networks with the main aim of identifying the most suitable caching policies. They conducted a series of simulations to investigate the different caching and replacement policies. Their results indicate that the best approach for IoT scenarios is a combination of the consumer cache caching strategy and Random Replacement (RR) cache replacement policy. This combination offers significant benefits in terms of reducing hops, decreasing server hits, and improving overall response latency [13].

Ikram Ud Din et al. presented a detailed analysis of different cache management strategies in the context of an ICN. Their study assessed the effectiveness of these strategies and identified their shortcomings. The evaluation is carried out through simulations in a network environment using cache hit rates, stretch ratios, and eviction operations as key performance metrics. They highlighted the persistent challenges in ICN caching that need to be addressed. Ultimately, their study provided a foundation for future research in the ICN domain with a focus on caching-related issues [14].

Muhammad Ali Naeem et al. Explored different caching strategies based on Information-Centric Networking (ICN). They examined various parameters, such as content retrieval time, cache success rates, path lengthening, and link load. They focused on IoT-based environments. Through a detailed simulation study, they demonstrated the significant benefits of ICN in-network caching, making it an essential asset for strengthening IoT networks [15].

Riya et al. Utilized popularity-based caching that prioritizes user preferences to reduce delays in fog computing. Their strategy involved organizing IoT devices into clusters based on their interests and proximity, using a method called spectral clustering. These clusters are then connected to fog nodes to enable efficient caching of frequently accessed files. To minimize latency when cache misses occur, a technique known as device-to-device (D2D) communication is used. Furthermore, association rules were utilized to predict future IoT device requirements. The findings of this experiment demonstrate that this approach is superior to other caching methods [16].

Emna Baccour et al. Extended the Collaborative Edge (CE) framework to include mobile video sharing (Device-to-Device - D2D) and introduced a new caching strategy to optimize the storage and transmission of video chunks through collaborative storage and sharing by MEC servers and users. They designed a D2D-aware proactive caching approach for frequently accessed content and formulated the problem as a linear program with suboptimal relaxation and an online heuristic to address NP-hardness. Through simulations, they demonstrated a performance improvement of over 10% compared to other caching methods in terms of hit ratio, delay, and cost [17].

Yingjie Duan et al. Introduced a reliable multicast framework based on ICN, with a focus on optimizing the recovery delay. To reduce the recovery delay, they proposed using a Congestion-Aware Probabilistic Cache (CAPC) strategy, which caches recently sent chunks during multicast. They also suggested using NACK feedback aggregation and recovery isolation techniques to reduce recovery overhead. Empirical results show that their approach is better than alternative methods in achieving dependable multicast with superior performance across recovery delay, cache hit ratio, transmission time, and overhead [18].

Emara M et al. Introduced a new mathematical approach that utilizes stochastic geometry to evaluate the likelihood of success in multicast 5G networks with caching capabilities. Their framework considered the multichannel capabilities of small base stations (SBS) and the ability to access the spectrum opportunistically. By assessing the success probabilities, service distance distributions, and coverage probabilities, they determined the hit probability. They also determined the most effective caching distribution for maximizing the hit probability and compared it with uniform and Zipf caching strategies using numerical analysis. Notably, Zipf caching is almost optimal, particularly in scenarios with sufficient channels and cache capacity [19].

Muhammad Ali Naeem et al. Proposed architecture allowed for clear in-network caching within networks, which improved content distribution and resource utilization. With increasing global network traffic, Content-Centric Network (CCN) caching has become a challenge. Therefore, flexibility in information retrieval is of utmost importance. The key similarity in CCN design is widespread caching, which enhanced performance by offering attributes, such as ubiquity, transparency, and content significance. Their work delved into contemporary CCN-based probabilistic caching methods, with a focus on reducing redundancy and improving content access [20].

Xiaoqing Chen et al. Developed a three-stage heuristic to improve the edge-cloud resource utilization for service caching and task offloading. Their approach prioritized cloud resources for task offloading and optimized edge resources for low latency. They also re-offload to the edge to improve the cloud task performance. Their simulations showed a significant improvement of up to 155% in user satisfaction, resource allocation, and processing efficiency [21].

III. INFORMATION-CENTRIC NETWORKING (ICN)

ICN is a new Internet architecture that transfers data based on the properties of the content being shared rather than opaque bytes and addresses. In an Information-Centric Networking (ICN) system, content objects are assigned unique names instead of IP addresses, making it possible to exchange content directly based on these names. This eliminates the need for a centralized system to map IP addresses to host names, which is currently required. This new approach offers greater flexibility for deployment across various network types, making it particularly useful for mobility. The unique features of in-network caching and replication indicate that content objects detach from their original locations. Many ICN architectures are designed from a clean slate perspective because several issues in the current Internet cannot be effectively resolved within its original design. Information exchange is a fundamental function of the Internet, and ICN architectures prioritize the discovery and transmission of information to end-users rather than the conventional model of transferring content from host to host. An information object can be located using its name rather than its address, which makes it easier to find and access the desired content. in (ICN) caching poses a distinctive challenge that requires effective content distribution and storage management to optimize performance and resource utilization.

A. ICN FOR IoT

The Growth in Internet traffic and evolving user expectations have led to the need for a new communication model. Users now want to access data from various devices and environments. Content Distribution Networks (CDNs) and peer-to-peer (P2P) overlays have been proposed as solutions to handle the increasing traffic, but they do not address fundamental issues with the current Internet architecture. Incremental solutions are also not sufficient to keep up with the evolution of the internet. To address these challenges, researchers and industry experts have been working on developing ‘‘Clean Slate’’ solutions that can offer a comprehensive resolution to the problems inherent in the existing internet architecture. Information-Centric Networking (ICN) is one such solution that has emerged as a promising option in IoT environments. ICN allows the requested content to be identified by its unique name and accessed from any cache holding the content. This approach has implications for the battery life of IoT devices, as a request can be fulfilled by an active node while the information producer remains in low-power sleep mode. The ICN also addresses security requirements by focusing on securing the contents themselves, rather than the channels connecting the devices. As a result, ICN enhances data dissemination and reduces network complexity, making it a great solution for IoT scenarios.

B. THE ADVANTAGES OF ICN-FOG LAYER

Let us discuss the benefits of ICN-Fog in more detail. Below are two real-world examples illustrating these advantages.

1. Reduced Dependency between Fog and Cloud: The ICN-Fog layer reduces the need for cloud support in applications. Instead of relying solely on the cloud to retrieve data, information can be served from the cache store of a local or nonlocal Fog Node (FN). Unlike conventional fog, ICN-Fog acts as a distributed collaborative cache pool, where only the cache store of the local/vertical FNs attached to the end device can be utilized.

2. Support for Diverse End Devices: ICN deployment in end devices is not required, freeing the ICN-Fog from the constraints imposed by the capabilities and resources of end devices. This flexibility enables a wider range of devices to participate because they only need to connect to local fog nodes. Fog nodes serve as proxies in the fog network for end devices, converting their requests into ICN requests, which are then forwarded within the ICN-fog layer or sent to the cloud. FNs store the data generated by end devices in their cache stores and publish them within the ICN-Fog network for future use.

The ICN prioritizes content over precise data locations, allowing for more efficient data retrieval and distribution. Fog computing places computing resources at the periphery of the network, reducing the distance that data must travel. This significantly decreases latency and enhances application responsiveness.

ICN’s content-centric approach of ICN, combined with the decentralized nature of fog computing, contributes to superior scalability, particularly in environments with numerous connected devices. The overall system reliability is enhanced by dispersing computing resources within the fog. In the event of a node failure, other nodes can continue their operations, mitigating the impact of such failures.

Utilizing the ICN’s caching mechanisms and the local processing capabilities of fog computing optimizes the bandwidth usage. Content can be cached and processed closer to the point of demand, leading to more efficient bandwidth utilization.

IV. CACHING

Caching involves storing duplicates of frequently accessed or calculated data to accelerate future access or computations. This is achieved by placing data in a cache, which is a smaller and faster storage location that allows for quick retrieval. Thus, the system can obtain data from the cache instead of recomputing or fetching it from the original source, resulting in faster processing times for future requests. When cache nodes reach their full capacity, content eviction schemes are used to free space for new incoming content. These schemes are typically adapted from the cache replacement algorithms used in operational system caching. To determine the content to evict, policies such as First-In-First-Out (FIFO), Least Frequently Used (LFU), and Least Recently Used (LRU) are employed based on factors such as content freshness, request frequency, and recency. Because most traditional caching schemes use LRU, which is based on linear speed, the proposed scheme also uses LRU to ensure fairness in the assessment.

A. ICN CACHING

Most Information-Centric Networking (ICN) approaches provide caching services to improve the speed of content access. Caching involves distributing multiple copies of a content object across a network. These replicas can be stored locally on an ICN node or shared across various network cache. Content caching is implemented at different network points, such as content routers, edge nodes, and end devices, and enables support for multi-sourcing for content consumers. The caching process operates at three levels: the object level (entire content object), chunk level (portion of a content object), and packet level (bytes of a content object).

B. CONTENT DELIVERY NETWORK (CDN)

A Content Delivery Network (CDN) is a network of servers that is distributed across various geographic locations. The primary purpose of this network is to ensure the swift delivery of Internet content by storing and caching it, thereby bringing the content closer to users. In the past, web content was delivered from a single server to all clients, which was not efficient in meeting the growing demands of today's content consumption, including high-resolution videos, music, graphics, and software. To overcome this challenge, content creators turn to CDNs to guarantee optimal user experience. CDNs play a vital role in overcoming the issues related to poor connectivity and content availability. Content caching in the current Internet architecture involves the following steps:

1. A user accesses the desired content through either a webpage link or an application.

2. If the content is not already in the browser cache or the user has not recently visited the web page, the browser connects to the nearest possible cache automatically, which is facilitated by techniques such as DNS resolution.

3. If the requested content is available in the cache, it is swiftly delivered to the user as the content becomes physically closer.

4. If the content is not in the cache, it is fetched from the original server and delivered to the user. While the initial request to the server may be slower, such occurrences are infrequent, ensuring consistently impressive retrieval times owing to content caching.

CDNs ensure that Internet content can be delivered directly to users from the nearest possible location, thereby reducing latency and load times. The cached content on a CDN server can benefit subsequent users requesting the same object from the same area, leading to faster load times. In addition, the CDN server acts as a buffer between the user and the original storage server, preventing the original server from being overwhelmed by a surge of direct requests. The distributed nature of CDN servers allows content to go viral without causing system crashes owing to heavy loads.

V. FIREFLY ALGORITHM

Nature-inspired algorithms and biological processes have recently been the most effective solutions for optimization problems. The Firefly Algorithm (FA) is an interesting concept inspired by the optical interactions observed among fireflies in their natural communication processes. This algorithm can be seen as a manifestation of swarm intelligence, in which the collaboration of less intelligent members leads to higher collective intelligence. Fireflies emit mesmerizing flashes of light as a means of communication, primarily to attract potential mates. The core idea behind the firefly algorithm is to mimic this flashing behavior to draw fireflies towards brighter ones, as shown in (Algorithm 1). Yang developed the original formulation of the firefly algorithm based on certain assumptions about the behavior of fireflies [22], [23]. The algorithm is based on the following three hypotheses.

1. Gender-independent attraction: Fireflies are attracted to each other irrespective of gender.

2. Attractiveness proportional to brightness: The less bright firefly is drawn towards the brighter one, with brightness decreasing as distance increases. If no brighter firefly is present, it moves randomly.

3. Brightness determined by the objective function: The brightness of a firefly is determined by the value of the objective function in the context of optimization.

Optimization algorithms aim to determine solutions in the search space that minimize or maximize the objective function. The complexity of an optimization problem is characterized by the mathematical relationships between the objectives, constraints, and decision variables. Fireflies in the Firefly Algorithm operate independently, making them suitable for parallel processing.

When attempting to maximize the efficiency in optimization problems, the brightness of a firefly at a specific position x is represented by $I(x) \propto f(x)$. However, the attractiveness factor β is subjective and depends on the observer's perspective or judgment of the other fireflies. Consequently, it fluctuates based on the distance r_{ij} between the i^{th} and j^{th} fireflies. In addition, because light intensity decreases with distance from the source and absorption in the medium, it is necessary to allow the appeal to adapt to the absorption level.

Algorithm 1 Pseudo Code of the Basic FA

```
1:Begin
2:Initialize algorithm parameters:
3:MaxGen: the maximum number of generations
4:  $\Upsilon$ : the light absorption coefficient
5:  $\beta_0$ : initial brightness of a firefly
6:  d : the domain space
7: Define f(X), where $X = (x_1, x_2, \ldots, x_d)^T$
8:Generate the initial population of fireflies, $X_i (i = 1,2,\ldots,n)$
9:Determine Light intensity of I, at $i^{th}$ firefly $x_i$ via $f(X_i)$.
10: while (t < MaxGen) do
11:     for i = 1: n (all n fireflies) do
12:         for j = 1: n (all n fireflies) do
13:             if ($I_j > I_i$) then
14:                 Move firefly i towards j
15:             end if
16:             Attractiveness varies with distance r via $e^{(-\gamma r^{\wedge}2)}$
17:             Evaluate new solutions and update light intensity.
18:         end for j
19:     end for i
20:     Rank the fireflies and then find the current best.
21: end while
22: Postprocess results and visualization.
```

For a given medium with a constant light absorption coefficient γ, the light intensity (I) changes with the distance (r) according to (1):

$$I = \mathbf{I_0} \times \mathbf{e}^{(-\gamma \times \mathbf{r}^{\wedge}\mathbf{2})} \tag{1}$$

where I_0 represents the initial light intensity. Because a firefly's allure depends on the light intensity perceived by neighboring fireflies, we can define the appeal β of a firefly as shown in (2):

$$\boldsymbol{\beta}(\mathbf{r}) = \boldsymbol{\beta_0} \times \mathbf{e}^{(-\gamma \times \mathbf{r}^{\wedge}\mathbf{2})} \tag{2}$$

The distance between two fireflies i and j located at x_i and y_j, respectively, is the Cartesian distance as shown in (3)**:**

$$\mathbf{r_{ij}} = ||\mathbf{x_i} - \mathbf{y_j}|| = \sqrt{\sum\nolimits_{\mathbf{k=1}}^{\mathbf{d}} (\mathbf{x_{i,k}} - \mathbf{y_{j,k}})} \tag{3}$$

where $\mathbf{x_{i,k}}$ and $\mathbf{y_{j,k}}$ are the k^{th} component of the spatial coordinate x_i and y_j of i^{th} and j^{th} fireflies, respectively.

A firefly i that is attracted to a brighter and more attractive firefly j moves according to the equation (4):

$$\mathbf{new(x_i)} = \mathbf{old(x_i)} + \boldsymbol{\beta_0} \times \mathbf{e}^{(\gamma \times \mathbf{r_{ij}}^{\wedge}2)}(\mathbf{y_j} - \mathbf{x_i}) + \boldsymbol{\alpha}(\mathbf{rand} - \mathbf{0.5}) \tag{4}$$

where the second term represents attraction and the third term adds randomization, where α represents the randomization parameter. Variable rand is a uniformly distributed random number in the range [0, 1].

Light intensity updated after each iteration, represented by eq (5):

$$\mathbf{L_i(t)} = (\mathbf{1} - \boldsymbol{\delta}) \times \mathbf{L_i(t-1)} + \boldsymbol{\Upsilon} \times \mathbf{L_i(t)} \tag{5}$$

where δ represent the rate of decreasing luciferin over time and Υ represent the noise (light absorption).

VI. PROPOSED MODEL OF (MOFA)

This proposed strategy involves utilizing fog nodes to temporarily store data. The demand for specific content originates from a group of users. The caching nodes nearer to the user devices are referred to as lower-stratum caches, while those closer to the core network are known as upper-stratum caches. The system model for the fog-caching scheme is described using both permanent character assignment and temporary role allocation. When a user device generates a new content request, the system identifies and records attributes such as caching capability as permanent characteristics to establish a node's capabilities for a delivery session. Each caching node acts as a virtual caching point that combines the node identity (ID) and stratum (tier) number. These pieces of information are utilized to allocate cache and determine which node will store the requested content.

A. TOPOLOGY DESIGN

Our network topology was designed in the form of a traditional tree structure, as illustrated in FIGURE 1. The central node n_1 serves as the core network hosting the source of the original content. The descendants of the root nodes n_2 and n_3 act as cloud-edge nodes that are connected to the outer edge of the fog. In our implementation, we do not consider the hops between the outer edge of the fog and the cloud edge, as they do not play any role in the proposed Fog-based scheme. Within the Fog, ICN-fog nodes (n_7 to n_{12}) are connected to access points (n_4 to n_6), which have forwarding capabilities. Finally, end-user devices (n_{13} to n_{19}) do not possess caching capabilities.

B. PERMANENT CHARACTER ASSIGNMENT

In the system concept presented, the structure of the Fog network is represented as a topology graph labeled as **G = (N, L).** Here, **N** corresponds to the collection of vertices within the graph (also referred to as nodes), while **L** symbolizes the connections between these nodes (graph edges). To be more precise, every node, denoted as n_i and belonging to the set **N**, possesses a distinct identifier **'I'.** Each n_i node is potentially associated with one or multiple categories from these subsequent sets:

1) SET OF USER GADGETS, UG

Encompasses user devices that start content delivery sessions by transmitting content request messages.

2) SET OF TRANSMITTING NODES, TN

Incorporates elements with the ability to transmit messages.

3) SET OF DATA ORIGINATOR, D-ORIG

Incorporates entities that generate content originally.

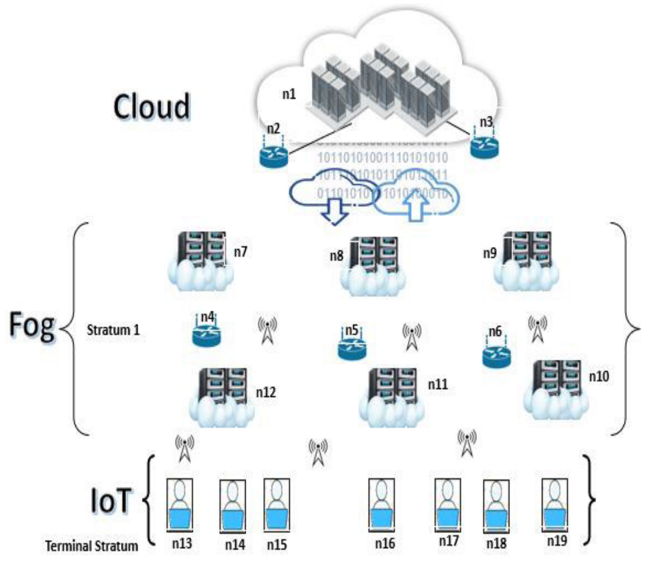


FIGURE 1. Proposed system architecture for content caching placement in fog computing.

TABLE 1. Characteristics of nodes.

Character	Node Set
User Gadget	UG=$\{n_i\}$, i ϵ [13-19]
Transmitting Node (ICN Routers)	TN=$\{n_i\}$, i ϵ [2-6]
Data Originator	D-Orig = $\{n_1\}$
Caching Node	CN=$\{n_i\}$, i ϵ [7-12]

4) SET OF CACHING NODES, CN

Comprising of nodes equipped with caching capabilities, these nodes could function as either individual caching nodes or ICN routers.

C. TEMPORARY ROLE ALLOCATION

In the topology of the fog network, the role of a node is determined by its position. In the illustrated example in FIGURE 1, the caching node n_3 is assigned the role of (1,3), indicating its location at stratum 1 with a node id of 3. Efficient fog caching is a crucial component of fog computing, as it involves the placement of data caches at edge nodes to minimize data access delay and enhance overall system performance. To optimize the cache placement methods, nature-inspired algorithms such as the Firefly Algorithm have been integrated and proven to be highly effective.

VII. SYSTEM IMPLEMENTATION

The proposed scheme involved a three-step implementation system. This includes preparation (STEP 1), Request Message Routing and Delivery (STEP 2), and Caching Content in the most efficient node for future use (STEP 3).

A. STEP 1. (PREPARATION)

The proposed model was constructed based on individual node character assignment and role allocation. The preparation step involves nominating the Bottom and Top caching nodes for request message routing (STEP 2) and content delivery and caching (STEP 3). Specifically, the original content provider node n_{pk} (where pk is the node's ID) is identified and recorded. Once the routing path from requesting end-user n_u to original content provider n_{pk} is established, the node for stratum 1 is identified along the delivery path and added to set Ct (1). The bottom caching node (n_{Bottom}) is the nearest caching node to the end user and the top cache node (n_{Top}) is the nearest caching node near the cloud server.

B. STEP 2. (REQUEST MESSAGE ROUTING AND DELIVERY)

If caching nodes are present in the Fog layer (denoted as Ct (1) $\neq \emptyset$) but a serving cache node has not been identified, the Cache hit variable is set to False. The request message then undergoes a traversal from the lowest to the highest node in search of a cache hit. The message is relayed from n_u to the bottom cache node(n_{Bottom}). During the fog tier traversal, the request message is sequentially broadcast from the bottom cache node(n_{Bottom}) to all nodes in the fog tier. If a cache hit occurs, the IDs record the provider ID and Sf is set to 1, indicating a cache hit in the first stratum. However, if no cache hits occur within the fog layer, the request message is forwarded to the original content source. Once the content is received from the content provider, Sf is assigned a value, and the IDs are set to the content provider's ID (pk). Subsequently, the caching process was initiated. In this scenario, the request routing and delivery process concludes, triggering the content delivery and caching process in STEP 3.

C. STEP 3. CONTENT DELIVER AND CACHING (THE PROPOSED MOFA FOR SELECTING NODE TO CACHE CONTENT)

When a user requests the content, the system searches for it. Once content is found, it is sent back to the user. In addition, the system stores a copy of the content in the fog layer using a caching approach that can be either reactive or proactive. This is possible owing to the full-time caching feature of the system. Reactive caching means that when the system delivers content K to user n_u, it places a single cache of K in the Fog Stratum. Proactive caching, on the other hand, aims to determine the most efficient node for cache content. If the content is found in the original content provider (Sf = 's') and there are nodes in the Fog layer (Ct $\neq \emptyset$), reactive caching is triggered to retain a cache in the Top cache node (n_{Top}). In this scenario, a copy of the content K is initially forwarded to the top cache node (n_{Top}) before being sent back to the user (n_u). For proactive caching, the system uses a nature-inspired optimization technique called the firefly algorithm. If a caching node exists in the fog stratum and the content request has already been fulfilled by the original content provider (Sf = 's'), a cache should already be present in the top cache node because of the earlier reactive caching process. A Multi-Objective Firefly Algorithm (MOFA) is proposed as a technique for executing proactive caching. MOFA aims to determine the most suitable fog node for caching content based on the efficiency of each fog node. The efficiency of each node is represented by the luciferin or light intensity of each firefly. The system selects the most suitable fog node to cache the requested content based on its position in the search space as shown in (ALGORITHM 2).

Algorithm 2 The Proposed MOFA for Selecting Node to Cache Content

```
Input Serving flag Sf, num fog nodes, serving node ID ids, requesting end-user device nu, num of contents, requested content, Data Originator $n_{pk}$, FC Ct (including $n_{Top}$), Cache size CS, Processing power P, Memory size M, Distance r, Attractiveness $\beta_0$, light absorption $\gamma$ and Max_Generation.
Output Finding the Most efficient node to cache content by applying (MOFA)
Define the Objective function f(x), $x = (x_1, \ldots, x_d)^T$
Generate the initial population of fireflies $x_i$ (i = 1, 2,3,.., f)
Light intensity $I_i$ (Efficiency of (i)fog node) at $x_i$ is determined by (6)
Define light absorption (Noise) coefficient $\gamma$.
1: If (Sf is 's') and (Ct $\neq \emptyset$) then
2:     Deliver content k from Data Originator $n_{pk}$ to Top
       cache node $n_{Top}$
3:     Put content k into Top cache node $n_{Top}$
4:     Forward content k from $n_{Top}$ to $n_u$
5: else
6:     Forward content k from serving node $n_{ids}$ to $n_u$
7: while (t < Max_Generation)
8:     for i = 1: f all f fireflies (caching nodes)
9:         Initialize luciferin (Efficiency of (i)fog node)
10:        for j = 1: f all f fireflies (caching nodes)
11:            Initialize luciferin (Efficiency of(j) fog node)
12:            if ($I_j > I_i$) then
13:            Move firefly i towards j
14:            end if
15:            calculate distance r between fireflies i and j
16:            $\beta \leftarrow \beta_0 e^{-\gamma rij}$ {obtain attractiveness}
17:            Evaluate innovative solutions and then
               update light Intensity.
18:        end for j
19:    end for i
20:    Rank the fireflies and then find the current best.
21:    Cache the requested content based on best selected
       firefly position in the search space.
22: end while
23: Postprocess results and visualization
24: return Most efficient node
```

VIII. EXPERIMENTAL SETUP

We used CloudSim to conduct experiments and evaluate the proposed in-network caching approach. The experiments were run on an Intel (R) Core (TM) i7-4510U with a 7-core CPU and 8GB RAM. In this section, we present a topology specifically designed to evaluate fog caching. The following subsections provide more details on parameter configuration and performance metrics.

TABLE 2. Available FOG servers.

FN_ID	M(GB)	CS(GB)	P(GHz)	D
N7	8	6	30	12
N8	16	2	50	4
N9	4	4	40	2
N10	3	5	40	6
N11	2	3	60	8

IX. SIMULATION PARAMETERS

Let's imagine a situation where we have five fog nodes, each with their own unique characteristics as outlined in TABLE 2. If we base our selection only on the criterion of processing power, then N11 would be the efficient node. On the other hand, selecting the largest memory size alone would designate N8 as the efficient node while going for the largest cache size would single out N7. If we consider the smallest distance, then N9 would emerge as the efficient node. Therefore, it is necessary to identify a single parameter that encapsulates all these characteristics within one equation and then determine its optimal value. This parameter is called efficiency (E), which quantifies the efficiency of each FN. Efficiency (E) is expressed in terms of processing power (P), memory size (M), cache size (CS), and distance (D), as shown in (6):

$$\boldsymbol{E} = \frac{\boldsymbol{M} \times \mathbf{CS} \times \boldsymbol{P}}{\boldsymbol{D}} \tag{6}$$

The FN with the highest E is likely to be chosen as the best solution to cache the content. FN operates with a Node Manager (NM) software, responsible for gathering information about each FN.

The arrival of request messages follows a Poisson distribution, which is commonly used in traditional caching experiments. The rate of content generation was set to 10 request messages per second. There are two main content parameters: the number of contents, denoted as |K|, and content popularity parameter, α. The number of contents varied based on the simulated scenario, and in our experiments, it amounted to [10-2000] contents. Content popularity distribution follows Zipf's law, represented as $1/(|K|^{\alpha})$, which is a commonly used choice. The α parameter within Zipf's law signifies the concentration of user preference and ranges from 0.1 to 0.9 in our experiments, with higher α values indicating more focused user preference. The caching parameters include network caching capacity, storage capacity allocation, and content eviction algorithm. the total network caching capacity to the number of contents |K|, as expressed in (7).

$$\mathrm{CC} = \frac{1}{|\mathrm{K}|} \sum\nolimits_{\mathrm{ci} \epsilon \mathrm{C}} |\mathrm{ci}| \tag{7}$$

Here, c_i represents the caching capacity of node i.

Each fog node assesses processing power (P), caching size (CS), memory size (M), and distance from the end user (D). Regarding the content eviction algorithm, our proposed scheme incorporates the Least Recently Used (LRU)

TABLE 3. Simulation parameters.

μ	**Transmission service rate.**
P_{TX}	**Transmission power.**
N_O	**Noise density.**
$\Upsilon_{(1,3,5...)}$	**Path loss exponent.**
$\Upsilon_{(2,4,6....)}$	**path loss constant.**
λ	**Arrival rate.**
B	**Bandwidth.**
g	**Gain channel.**
d	**Distance.**

algorithm, which is known for its constant time complexity and compatibility with numerous existing ubiquitous caching schemes. Moreover, determining fog efficiency requires assessing the processing power, memory size, caching size, and distance. The efficiency value serves to ascertain the luciferin value (light intensity) of each firefly (fog node), helping identify the optimal node for caching content for future use.

X. PERFORMANCE METRICS

There are three metrics used for performance evaluation, which are Average Query Duration (AQD), cache hit ratio (CHR) and internal link load (LL_{in}).

A. AVERAGE QUERY DURATION (AQD)

The average query duration refers to the time interval between the initiation of a query by the requester and the successful transmission of data back to the requester. The calculation of AQD involves utilizing equation (32), where T_D denotes the transmission delay (query duration) and R_{succ} represents the count of successfully processed queries.

In our framework, we interpret T_D as encompassing both the transmission delay (comprising both sending and response times) and the round-trip time (RTT) involved in relaying data fragments between end-user nodes, Fog Nodes (FNs) Cloud node. The transmission delay (T_D), can be evaluated using the M/D/1 system outlined in equation (8).

$$T_D = FN_{RT} + FN_{RPT} \tag{8}$$

Specifically, the transmission delay from an end-user node (e) to an FN (i) can be expressed as the FN request time (FN_{RT}) from e to i, as outlined in equation (9).

$$T_{ei} = \frac{\lambda_{ei}}{2\mu_{ei}(\mu_{ei} - \lambda_{ei})} + \frac{1}{\mu_{ei}} \tag{9}$$

Whereas

A) Transmission service rate from end-user nodes (e) and Fog Node (i), as outlined in equation (10).

$$\mu_{ei} = B_e \times \log_2 (1 + \frac{g_{ei} \times P_{TX,e}}{B_e \times N_o^e}) \tag{10}$$

B) Channel gain from end-user nodes (e) and Fog Node (i), as outlined in equation (11).

$$g_{ei} = \gamma_1 \times d_{ei}^{-\gamma_2} \tag{11}$$

Similarly, the FN response time (FN_{RPT}) from i to e is defined by equation (12).

$$T_{ie} = \frac{\lambda_{ie}}{2\mu_{ie}(\mu_{ie} - \lambda_{ie})} + \frac{1}{\mu_{ie}} \tag{12}$$

Whereas

A) Transmission service rate from Fog Node (i) and end-user nodes (e) is as eq (13).

$$\mu_{ie} = B_i \times \log_2 (1 + \frac{g_{ie} \times P_{TX,i}}{B_i \times N_o^i}) \tag{13}$$

B) Channel gain from Fog Node (i) and end-user nodes (e) is as eq (14).

$$g_{ie} = \gamma_3 \times d_{ie}^{-\gamma_4} \tag{14}$$

Hence the transmission delay (T_{D1}) from end-user nodes (e) and fog node is as eq (15).

$$\begin{aligned} T_{D1} &= T_{ei} + T_{ie} \\ &= \frac{\lambda_{ie}}{2\mu_{ie}(\mu_{ie} - \lambda_{ie})} + \frac{1}{\mu_{ie}} + \frac{\lambda_{ie}}{2\mu_{ie}(\mu_{ie} - \lambda_{ie})} + \frac{1}{\mu_{ie}} \end{aligned} \tag{15}$$

Similarly, the transmission delay between FN (i) and neighboring node (j) can be expressed as the FN request time (FN_{RT}) from i to j, as outlined in equation (16).

$$T_{ij} = \frac{\lambda_{ij}}{2\mu_{ij}(\mu_{ij} - \lambda_{ij})} + \frac{1}{\mu_{ij}} \tag{16}$$

Whereas

A) Transmission service rate from Fog Node (i) to neighboring node (j), as outlined in equation (17).

$$\mu_{ij} = B_i \times \log_2 (1 + \frac{g_{ij} \times P_{TX,i}}{B_i \times N_o^i}) \tag{17}$$

B) Channel gain from Fog Node (i) to neighboring node (j), as outlined in equation (18).

$$g_{ij} = \gamma_5 \times d_{ij}^{-\gamma_6} \tag{18}$$

Likely, the FN response time (FN_{RPT}) from j to i is defined by equation (19).

$$T_{ji} = \frac{\lambda_{ji}}{2\mu_{ji}(\mu_{ji} - \lambda_{ji})} + \frac{1}{\mu_{ji}} \tag{19}$$

Whereas

A) Transmission service rate from Fog Node (j) to node (i), as outlined in equation (20).

$$\mu_{ji} = B_j \times \log_2 (1 + \frac{g_{ji} \times P_{TX,j}}{B_j \times N_o^j}) \tag{20}$$

B) Channel gain from Fog Node (j) to node (i), as outlined in equation (21).

$$g_{ji} = \gamma_7 \times d_{ji}^{-\gamma_8} \tag{21}$$

Hence the transmission delay (T_{D2}) from fog node (i) to neighboring (J) is as eq (22).

$$T_{D2} = T_{ij} + T_{ji}$$

$$= \frac{\lambda_{ij}}{2\mu_{ij}(\mu_{ij} - \lambda_{ij})} + \frac{1}{\mu_{ij}} + \frac{\lambda_{ji}}{2\mu_{ji}(\mu_{ji} - \lambda_{ji})} + \frac{1}{\mu_{ji}} \quad (22)$$

While searching nodes in fog layer the total transmission delay T_{D2}^{Total} is determined by eq(23).

$$T_{D2}^{Total} = \sum_{i=n_{Bottom}}^{j=n_{Top}} T_{ij} + T_{ji} \quad (23)$$

Similarly, the transmission delay between Cloud(c) and neighboring node (j) can be expressed as the FN request time (FN_{RT}) from j to c, as outlined in equation (24).

$$T_{jc} = \frac{\lambda_{jc}}{2\mu_{jc}(\mu_{jc} - \lambda_{jc})} + \frac{1}{\mu_{jc}} \quad (24)$$

Whereas

A) Transmission service rate from Fog Node (j) to cloud node (c) is determined by eq(25).

$$\mu_{jc} = B_j \times \log_2(1 + \frac{g_{jc} \times P_{TX,j}}{B_j \times N_o^j}) \quad (25)$$

B) Channel gain from Fog Node (j) to cloud node (c) is determined by eq(26).

$$g_{jc} = \gamma_7 \times d_{jc}^{-\gamma_8} \quad (26)$$

Likely, the FN response time (FN_{RPT}) from cloud node (c) to fog node (j) is defined by equation (27).

$$T_{cj} = \frac{\lambda_{cj}}{2\mu_{cj}(\mu_{cj} - \lambda_{cj})} + \frac{1}{\mu_{cj}} \quad (27)$$

Whereas

A) Transmission service rate from cloud node (c) to fog node (j) is determined by eq(28).

$$\mu_{cj} = B_{cj} \times \log_2 \left(1 + \frac{g_{cj} \times P_{TX,c}}{B_c \times N_o^c}\right) \quad (28)$$

B) Channel gain from cloud node (c) to fog node (j) is determined by eq(29).

$$g_{cj} = \gamma_7 \times d_{cj}^{-\gamma_8} \quad (29)$$

Hence the transmission delay (T_{D3}) from cloud node (c) to fog node (j) is determined by eq(30).

$$\begin{aligned} T_{D3} &= T_{jc} + T_{cj} \\ &= \frac{\lambda_{jc}}{2\mu_{jc}(\mu_{jc} - \lambda_{jc})} + \frac{1}{\mu_{jc}} + \frac{\lambda_{cj}}{2\mu_{cj}(\mu_{cj} - \lambda_{cj})} + \frac{1}{\mu_{cj}} \end{aligned} \quad (30)$$

As a result, the total transmission delay (T_D) is calculated as in eq (31).

$$T_D = T_{D1} + T_{D2}^{Total} + T_{D3} \quad (31)$$

It is important to note that, if content found in fog node(i)

$$\longrightarrow T_{D2}^{Total}, \quad T_{D3} = 0.$$

Else If content found in (j) $\longrightarrow T_{D3} = 0$

$$\text{Else} \longrightarrow (T_D = T_{D1} + T_{D2}^{Total} + T_{D3}).$$

Finally, the (AQD) will determined by eq (32)

$$AQD = \frac{\sum_{R=0}^{Rsucc} (T_D)}{Rsucc} \quad (32)$$

where Rsucc is the total number of successful queries. T_D is transmission (Query) delay time.

B. CACHE HIT RATIO (CHR)

In caching, the cache hit ratio is a measure of how many requests are fulfilled by cache nodes instead of servers. Let us say that there are M requests being processed and the entities that serve them are either content sources or fog caches. When a cache containing the requested content is found on one of the fog nodes within the FC, it is called a cache hit and registered in the H_C counter. However, if the request can't be met within the FC and is instead fulfilled by the original content source, it's recorded as a server hit H_s. The cache hit ratio was calculated using eq (33).

$$CHR = \frac{R_{Hit}}{R_{Total}} = \frac{H_C}{H_c + H_s} = \frac{H_c}{|M|} \quad (33)$$

C. INTERNAL LINK LOAD (LL_{IN})

The average traffic load on the links within the fog network is denoted by LL_{in}. The link load (LL_e) is contributed by the quantities of contents ($N_o(e)$) and requests ($N_m(e)$) transmitted through link e, assuming that m is the message size of each request and o_k is the content size of the requested object k.so (LL_e) is calculated by eq (34).

$$LL_e = \frac{1}{T_D}[|m| \times N_m(e) + |o| \times N_o(e)] \quad (34)$$

Assume E_{in} refers to the set of internal links in the fog topology. The mean internal link load for the active links LLin, in the Fog topology was calculated according to eq (35).

$$LL_{in} = \frac{1}{E_{in}} \sum_{e \in E_{in}} LL_e \quad (35)$$

XI. RESULTS AND DISCUSSION

To compare different caching strategies, we selected four reference points: Consumer Caching (CSM), Leave Copy Down(LCD), Random Choice (RAND) and Full-Time Caching (FTC). CSM is a common approach that focuses on caching node placement, in which popular content is locally cached on end-user devices. LCD, on the other hand, is based on the traditional content caching strategy for Information-Centric Networking (ICN), where cached content is progressively moved towards end users in response to repeated requests. For random-based content placement, RAND caches content in a randomly chosen caching node along the delivery path. Finally, Full-Time Caching (FTC) first caches content in the highest tier through a reactive cache, then in the nearest caching node to the end user, and even on consumer devices with caching capabilities through proactive cache.

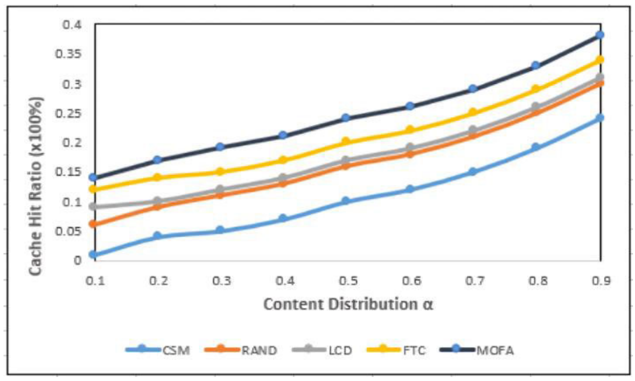


FIGURE 2. CHR of MOFA against benchmark schemes.

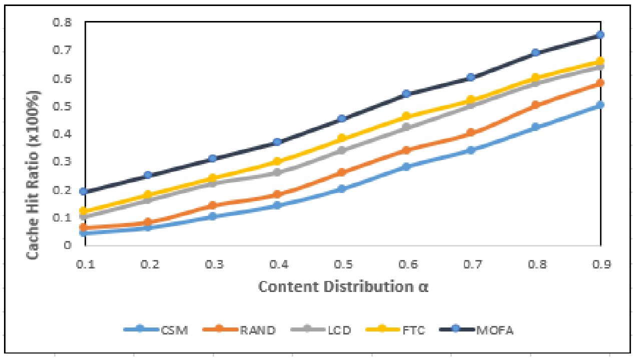


FIGURE 3. CHR of MOFA against benchmark schemes.

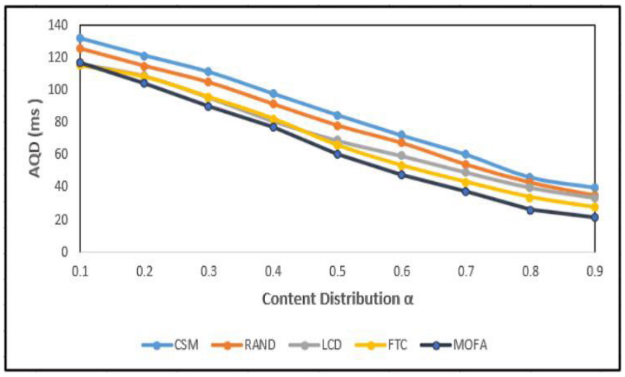


FIGURE 4. AQD of MOFA against benchmark schemes.

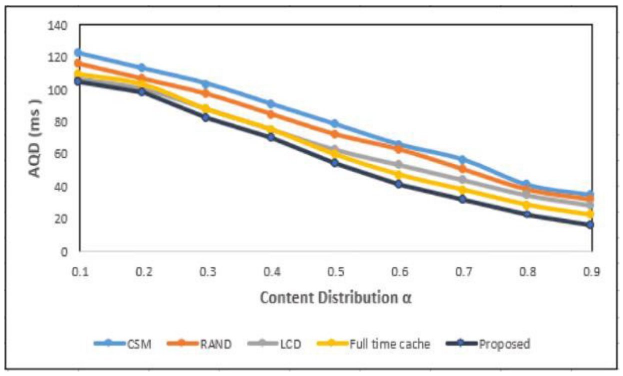


FIGURE 5. AQD of MOFA against benchmark schemes.

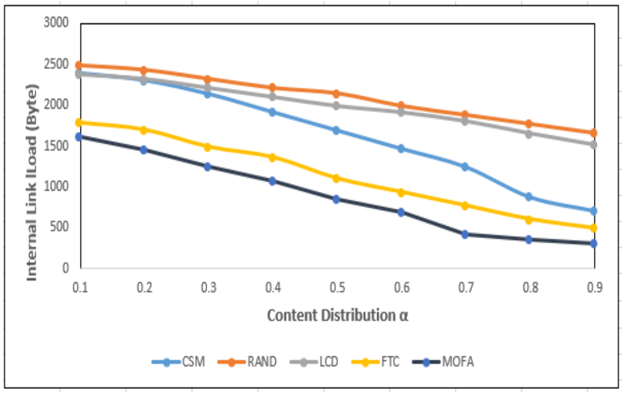


FIGURE 6. LLin of MOFA against benchmark schemes.

A. CACHE HIT RATIO (CHR)

Cache hit ratio is an important metric to consider when evaluating the three metrics. This is because a high cache hit ratio means that the core network has a lighter workload, and the content requests are evenly distributed among the available cache resources. Generally speaking, when the cache hit ratio is high, the caching nodes that are in close proximity to the user would be the ones to satisfy content requests rather than the distant original content source. To test our approach, we conducted experiments in which we systematically varied the skewness (α) and caching capacity (CC) parameters across all examined values. FIGURES 2 and 3 show the cache hit ratio performance for all five caching schemes under different α values from 0.1 to 0.9 while maintaining the CC at 0.2% and 5%, respectively.

B. AVERAGE QUERY DURATION (AQD)

The average query duration measures the time it takes for a request to be initiated by the requester, and for the data to be successfully transmitted back to them. FIGURES 4 and 5 show the performance of the Average Query Duration for all five caching schemes across different α values, with the CC remaining constant at 0.2% and 5%, respectively. As the network's CC increases, the latency naturally decreases owing to the greater capacity for content caching become more focused (indicated by a higher α value), the latency for all caching schemes decreases.

Essentially, the efficiency of caching improves as the content popularity becomes more concentrated, leading to increased α values.

C. INTERNAL LINK LOAD (LL_{IN})

The internal link load (LL_{in}) represents the average traffic load on links within the fog network. In FIGURES 6 and 7, the performance of the Internal Link Load for all five caching schemes is illustrated across various α values

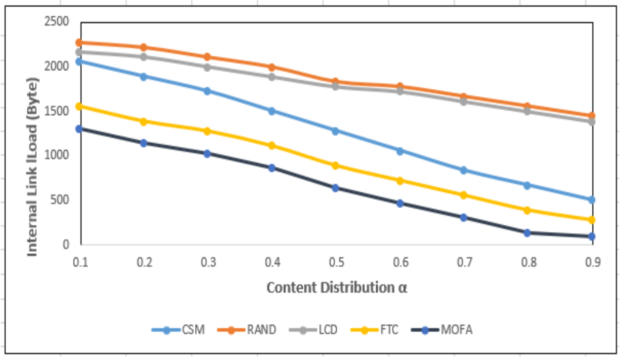


FIGURE 7. LLin of MOFA against benchmark schemes.

TABLE 4. Related and the proposed work comparison.

Strategy	Storage cost	Availability	Save bandwidth	Delay	Optimal location
CSM	√	×	×	√	×
RAND	√	√	×	√	×
LCD	√	√	√	√	×
MOFA	√	√	√	×	√

while maintaining the CC at 0.2% and 5%, respectively. When the CC of the network increases, the Internal Link Load decreases naturally, owing to the improved capacity for content caching in the network. Moreover, as user preferences become more concentrated (indicated by a higher α value), the (LL_{in}) for all caching schemes decreases. This means that caching efficiency improves as content popularity becomes more focused, leading to an increase in the α values.

The following table indicates the advantages of MOFA strategy against other strategies like CSM, RAND and LCD.

XII. CONCLUSION

Optimal availability and efficient data caching are essential components of cloud computing. In this study, we present a Multi-Objective Firefly Algorithm (MOFA) for caching that aims to improve caching by ensuring optimal access to frequently used data caches. The MOFA refines the placement of data caches using the firefly algorithm to correct their positions near users. We implemented the proposed system architecture using CloudSim and compared its effectiveness with other caching methods, such as LCD, RAND, CSM, and Full-time caching (FTC). The simulation results revealed that the MOFA-based algorithms outperformed the compared algorithms, indicating their increased efficacy. In future research, we plan to validate the proposed system architecture in a real cloud-computing environment. We also aim to address two pertinent caching issues: improving the precision and energy efficiency of complex data caches and utilizing the latest swarm intelligence dynamic data cache algorithms in cloud computing.

REFERENCES

[1] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, Sep. 2019.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st, Ed., MCC Workshop Mobile cloud Comput.*, Aug. 2012, pp. 13–16.

[3] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[4] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[5] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi, "A survey on content-oriented networking for efficient content delivery," *IEEE Commun. Mag.*, vol. 49, no. 3, pp. 121–127, Mar. 2011.

[6] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[7] Md. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 44–53, Dec. 2012.

[8] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Comput. Netw.*, vol. 57, no. 16, pp. 3128–3141, Nov. 2013.

[9] Y. Hua, L. Guan, and K. G. Kyriakopoulos, "A fog caching scheme enabled by ICN for IoT environments," *Future Gener. Comput. Syst.*, vol. 111, pp. 82–95, Oct. 2020.

[10] M. A. Naeem, Y. B. Zikria, R. Ali, U. Tariq, Y. Meng, and A. K. Bashir, "Cache in fog computing design, concepts, contributions, and security issues in machine learning prospective," *Digit. Commun. Netw.*, vol. 9, no. 5, pp. 1033–1052, Oct. 2023.

[11] I. Abdullahi, S. Arif, and S. Hassan, "Ubiquitous shift with information centric network caching using fog computing," in *Proc. 4th INNS Symposia Ser. Comput. Intell. Inf. Syst. (INNS-CIIS)*. Springer, 2015.

[12] D. Nguyen, Z. Shen, J. Jin, and A. Tagami, "ICN-fog: An information-centric fog-to-fog architecture for data communications," in *Proc. IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–6.

[13] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, and K. Drira, "How to cache in ICN-based IoT environments?" in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2017, pp. 1117–1124.

[14] I. U. Din, S. Hassan, M. K. Khan, M. Guizani, O. Ghazali, and A. Habbal, "Caching in information-centric networking: Strategies, challenges, and future research directions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1443–1474, 2nd Quart., 2018.

[15] 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 Things J.*, vol. 9, no. 9, pp. 6477–6495, May 2022.

[16] N. Gupta and S. K. Dhurandher, "Efficient caching method in fog computing for Internet of Everything," *Peer-Peer Netw. Appl.*, vol. 14, no. 1, pp. 439–452, Jan. 2021.

[17] E. Baccour, A. Erbad, A. Mohamed, M. Guizani, and M. Hamdi, "Collaborative hierarchical caching and transcoding in edge network with CE-D2D communication," *J. Netw. Comput. Appl.*, vol. 172, Dec. 2020, Art. no. 102801.

[18] Y. Duan, H. Ni, and X. Zhu, "Reliable multicast based on congestion-aware cache in ICN," *Electronics*, vol. 10, no. 13, p. 1579, Jun. 2021.

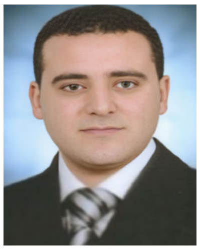
[19] M. Emara, H. ElSawy, S. Sorour, S. Al-Ghadhban, M.-S. Alouini, and T. Y. Al-Naffouri, "Optimal caching in multicast 5G networks with opportunistic spectrum access," in *Proc. IEEE Global Commun. Conf.*, Dec. 2017, pp. 1–7.

[20] 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.

[21] X. Chen, T. Gao, H. Gao, B. Liu, M. Chen, and B. Wang, "A multi-stage heuristic method for service caching and task offloading to improve the cooperation between edge and cloud computing," *PeerJ Comput. Sci.*, vol. 8, p. e1012, Jun. 2022.

[22] S. Lukasik and S. Zak, "Firefly algorithm for continuous constrained optimization tasks," in *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, Wroclaw, Poland. Berlin, Germany: Springer, 2009.
[23] X.-S. Yang, "Multiobjective firefly algorithm for continuous optimization," *Eng. Comput.*, vol. 29, no. 2, pp. 175–184, Apr. 2013.

MOHAMED R. ELNAGAR received the B.Sc. degree in information systems from the Faculty of Computers and Informatics, Suez Canal University, Ismailia, Egypt, in 2019. He is currently a M.Sc. Researcher with the Faculty of Computers and Informatics, Suez Canal University.


AHMED AWAD MOHAMED received the Ph.D. degree in information system from Menoufia University, Egypt, in 2021. He is currently a Lecturer with the Information Systems Department, Cairo Higher Institute for Languages and Simultaneous Interpretation, and Administrative Science. Furthermore, he has been reviewing articles for 18 international journals, including *CMC-Computers, Materials and Continua* and *Expert Systems with Applications*. Also, he was registered as a four patent in Egyptian patent office. He has been invited as a guest in many TV programs to comment on inventions. His major interests include distributed systems, the IoT, optimization, cloud computing, and big data. Besides, he is an Egyptian Inventors Syndicate (EIS) Member.

BENBELLA S. TAWFIK received the B.Sc. and M.Sc. degrees from the Military Technical College, in 1986 and 1990, respectively, and the Ph.D. degree from Colorado State University, in 1998. He is currently a Professor with the Faculty of Computers and Informatics, Suez Canal University, Egypt. He has more than 30 years of experience in computer science field. His research interests include computer networks, wireless sensor networks, information systems, pattern recognition, and image processing.

HOSAM E. REFAAT received the degree from the Faculty of Science, Assuit University, Egypt, in 1998, and the master's degree in distributed systems from the Faculty of Science, Cairo University, Egypt, in October 2006. He is currently an Associate Professor with the Faculty of Computers and Informatics, Suez Canal University, Egypt. His current research interests include parallel systems, cloud computing, edge computing, and data mining.