Journal of Theoretical and Applied Information Technology
15th October 2019. Vol.97. No 19

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A REVIEW OF CACHING STRATEGIES AND ITS CATEGORIZATIONS IN INFORMATION CENTRIC NETWORK

MOHAMMAD ALKHAZALEH[1], S. A. ALJUNID[1], NASEER SABRI[1]
[1]Advanced Communication Eng., Centre of Excellence, School of Computer and Communication Eng.
University Malaysia Perlis

ABSTRACT

Information-centric networking (ICN) is one of the rising Internet paradigms proposed to beat the defect of the current host-centric Internet. ICN relies on the name to access contents rather than its original location, which provides substantial flexibility for users to obtain the contents. In-network caching is one of the most important features of the ICN because it has a significant role in improving network performance such as reducing server load, congestion, and delay caused by users. In-network caching is managed by a caching strategy that determines what, where and when to cache the content, to make content is available for requesters without going to servers. In recent years, caching strategies have attracted considerable interest from researchers, and they have proposed many caching strategies to manage the contents to enhance the performance of ICN. In this paper, the caching strategies and its characteristics are clearly described and discussed; furthermore, the caching strategies are categorized based on a set of common characteristics to be easy to understand.

Keywords: *Information-Centric Networking, Content Caching Strategies, In-network Caching, Caching Mechanisms.*

1. INTRODUCTION

Due to the rapid growth of the Internet, the current P-2-P system has many drawbacks when it deals with a large scale of content distribution[1]. Nowadays, users are interested in accessing content other than their original location [2], especially a video that is widely used these days. For that, a new infrastructure named ICN has been proposed as an alternative to the current Internet.

The main idea about ICN is to access the content by the name other than the original location [3][4]. There are many architectures built on this idea and all fall under the ICN but differ in the implementation such as NDN[5], CCN[6], DONA[7].

ICN includes in-network caching[8], which is one of the main advantages of ICN. In-network caching improves network performance such as reducing server load, congestion, and delay caused by users [9] [10]. In-network caching allows to cache copies of content on routers in different locations, In order to make this content closer to the user. The caching strategy is responsible for managing the contents and selecting the appropriate location to cache a copy of content, and the performance of the in-network caching depends on a caching strategy, which consists of two parts, placement which chooses the appropriate location to cache a copy of the content, and replacement is responsible for evicting the content from the cache when it is full and new content has come to store in this cache [11].

In recent years, the researchers have been interested in ICN and especially the In-network caching, and have introduced several strategies for managing the contents in the Cache, but these caching strategies differ in term cache decision. For that, at the beginning of this review will be clarified the In-network caching feature in ICN and its importance. Then the categorizations of caching strategies will be defied, which based on some shared attributes between the principles of caching strategy to take decision to cache the content. After that, a comprehensive review of caching strategies will be introduced, which explain and summarize the existing caching strategies to be easy to understand. In addition to mentioning the modern caching strategies that have not been mentioned in the previous review papers such as MAGIC [12],

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

BidCache [13], LCLCS [14], APP [15], and PGBCS [16]. Then, the caching strategies will be classified based on these categorizations to realize and understand what are the similarities and differences between the principles of caching strategies to take decision to cache the content, which will be described in the following lessons:

2. CACHING IN ICN

ICN includes in-network caching features at every content router [8][11], which allow content to be cached at some of the content routers in the network, therefore provide efficient data delivery[17]. The caching consider one of the main advantages of the ICN, it allows to users and content routers to satisfy locally without going to servers, when consumer generate request for content, at any content router must cache the content to support other users and content routers maybe request the content and satisfy locally, this mechanism named placement, also which content to be replaced first when the cache is full, this mechanism named replacement [18][19].

Caching in ICN is an important thing because the efficiency of ICN based on caching strategies that manage the contents distribution along networks, which make the contents are available when the user requests these contents. The researchers proposed many caching strategies to manage the content distribution among the networks, but these caching strategies differ in term cache decision, some of these strategies based on the path, and other on probability, popularity, corporation, and content type.

3. CATEGORIES OF CACHING MECHANISMS

The objectives of caching mechanisms are to distribute content copies on the delivery paths, minimize content delivery time, and lessen the number of replicas. The current caching mechanisms are organized into several categories according to their attributes.

3.1 Path Caching:

Caching mechanisms can be divided into off-path caching and on-path caching[20]. For off-path, a Data packet is not necessarily to cache at the content router across the path to the requester. In contrast, a Data packet is cached across that path in on-path caching.

3.2 Redundancy Caching

Redundancy caching refers to a situation where a content is cached at more than one content router between requester and server (downloading path). Otherwise, it is a non-redundancy caching.

3.3 Cooperation Caching

Cooperative caching refers to the situation where content is cached according to the information provided by other content routers. It can be classified into cooperation implicit and explicit cooperation. With implicit cooperation, the delivery of content routers' cache states through an extra advertisement mechanism is not necessary. Nonetheless, additional operations are needed to assist content routers in making forwarding or caching decision. In explicit cooperation, if a content router holds a copy of content, it has to advertise its cache state to other content routers.

3.4 Popularity Caching

The Popularity is the number of request for content at the content router, some of the caching strategies based on popularity to take the decision to cache content, and others don't do that, for that, the caching strategies can be divided into Popularity Caching and Non-Popularity caching.

3.5 Distribution Caching

Some of caching strategies based on distribution, as the distance between the content router and the server or content router position. For that, caching strategies can be divided into Distribution Caching and Non- Distribution Caching.

3.6 Probability Caching

Some of caching strategies select content router to cache content based on content Probability, for that, caching strategies can be divided into Probability Caching and Non- Probability caching.

3.7 Content Type Caching

Every caching strategy has used Content type as, packet, chunk object, file and content, the packet and chunk consider same, also object and file and content consider same, for that, caching strategies based on Content type can be classified into chunk caching and content caching.

4. EXISTING CACHING MECHANISMS

This section explains the current caching mechanisms and the decision-making process on content caching.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.1 Leave Copy Everywhere (LCE) [21].

For CCN/NDN, the default caching mechanism is LCE. LCE aims to cache the Data-packet identical to the Interest-packet sent by the requester in every content-router across the line from the content-router responding to the Interest packet to the requester, as shown in Figure 1.

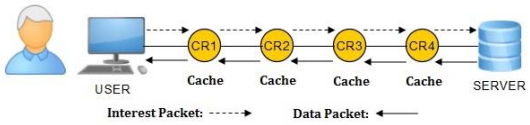

Figure 1: LCE operations.

4.2 Leave Copy Down (LCD) [21].

The objective of LCD is to reduce the caching redundancy by including a caching proposal flag bit in the header of Data-packet. The flag bit will be activated when the content source or content router with a replica of conforming content receives a request. Using this bit, the content-routers across the transferring path will evaluate the necessity to cache the Data-packet.

In LCD, a copy of content will leave at the content router in one level down towards the requester after each request, as shown in Figure 2.

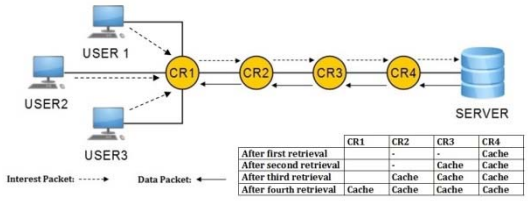

Figure 2: LCD operations.

4.3 Move Copy Down (MCD) [21].

The approach employed by MCD in minimizing the caching redundancy is similar to LCD. In MCD, the copy of content will move at the content router in one level down towards the requester after each request, as shown in Figure 3.

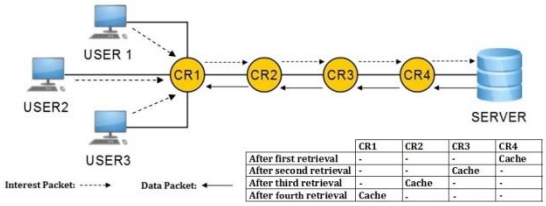

Figure 3: MCD operations.

4.4 Caching with Probability (Prob(p)) [21].

The rationale of caching with probability is to reduce caching redundancy and enhance efficiency. With probability *p*, every content-router in Prob(p) will cache a Data-packet. However, for a likelihood of 1-*p*, the Data-packet will not be stored. A random number in a range of zero to one will be produced once a Data-packet arrives at a content-router. The content-router will cache a copy of the Data-packet if the random number generated is less than *p*. Otherwise, as demonstrated in Figure 4, the Data-packet will be forwarded without being cached.

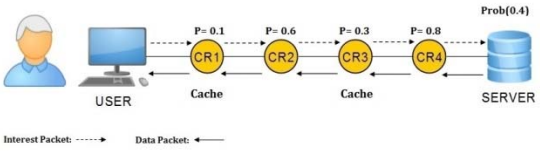

Figure 4: Prob(p) operations.

4.5 Breadcrumbs [22].

Figure 5 shows the caching mechanisms of Breadcrumbs. Along the downloading path, a Data-packet is stored at every content-router. When the Data-packet is cached, a data routing history, also known as a caching-trail is created by a content-router. This trail includes content name, forwarding time, upstream, and downstream. Upstream indicates the origin of a incoming Data-packet at the content router, while downstream indicates the destination of an outgoing Data-packet from the content router.

For every caching trail, Breadcrumbs assign a threshold denoted as Tha. The next hops of Interest-packets will be decided based on the content name's caching-trail at a content-router. When an Interest-packet is received by a content-router nonetheless a cached copy in its Content Source:

If (Interest-packet time – Forwarding Time) <Tha
Then downstream
Else upstream.

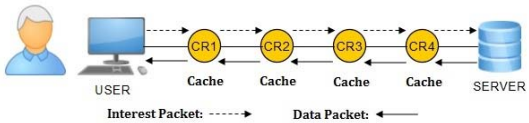

Figure 5: Breadcrumbs operations.

4.6 Probcache [23].

In Probcache, the probabilistic caching mechanism includes a Time-Since-Inception (TSI) part in the header of Interest-packet and a Time-

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Since-Birth part in the header of Data-packet. Initially, the TSI cost is at zero once an Interest-packet is sent out by a user. After each hop, the value will increase by one. On the other hand, the TSB value is set at zero initially as soon as a Data-packet is sent out by a content source. After that, the value will increase by one per hop. Nonetheless, the final value of TSI remains unchanged.

TimesIn (T_In) refers to the sum of times that a packet can be saved on a path. Using *TimesIn*, the available caches on the downloading path can be estimated.

TSI value contained in the Interest-packet field header and TSB cost contained in the header of a Data-packet are denoted by *y* and *z*, respectively.

T_{tw} value refers to the duration that a content must be stored on a distribution pathway such as 10.

For the distribution pathway, the cache size of content-router n, Ni, is represented by the duration of cached content maintained in a cache. The mean of the cache size of a distribution pathway, N_z, is determined by the cache size of the content-routers on a delivery path.

$$T\_In(z) = \frac{\sum_{i=1}^{y-(z-1)} N_i}{T_{tw} N_z} \quad (1)$$

$$CacheWeight(\mathrm{z}) = \frac{z}{y} \quad (2)$$

$$Probcache(\mathrm{z}) = T\_In(z) * CacheWeight(\mathrm{z}) \quad (3)$$

Probcache calculates the TimesIn and the cache weight, then calculates caching probability (ProbCache(z)) at each content router by using equation 3, a content router with a big (ProbCache(z)), compared to the content routers along the path, will be caching contents with higher probability as shown in Figure 6.

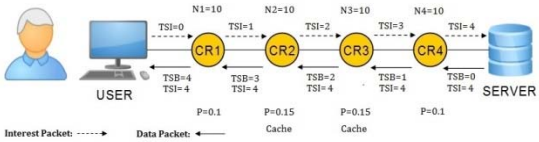

Figure 6: Probcache operations.

4.7 Probcache+ [23].

ProbCache+ is an enhanced version of ProbCache aim to reasonable distribution of volume resources on a distribution pathway amongst contents. It has the same variables but there is a slight difference in calculations. That is illustrated in Figure 7.

$$T\_In(z) = \frac{\sum_{i=1}^{y-(z-1)} N_i}{T_{tw} N_z} \quad (4)$$

$$CacheWeight(\mathrm{z}) = \left(\frac{z}{y}\right)^{z} \quad (5)$$

$$Probcache(\mathrm{z}) = T\_In(z) * CacheWeight(\mathrm{z}) \quad (6)$$

Probcache+ calculates (ProbCache(z)) at each content router like the Probcache, a content router with a big (ProbCache(z)), compared to the content routers along the path, will be caching contents with higher probability as shown in Figure 7.

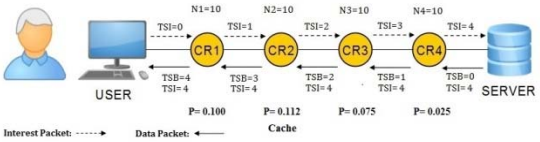

Figure 7: Probcache+ operations.

4.8 WAVE [24].

The purposes of WAVE are to ensure effectual content distribution and cache exploit; and to reduce the onus of the cache administration. WAVE work similar to LCD but use chunk of content, it includes a caching-suggestion flag bit in Data-packets to assist content-routers in making decisions on caching.

Within ICNs, a sizeable packet is split amongst various chunks. For instance, a content CX is split accordingly into seven chunks, namely CX1, CX2, CX3, CX4, CX5, CX6 and CX7. To obtain CX, a requester will issue seven Interest-packets that correspond to respective chunks of the content.

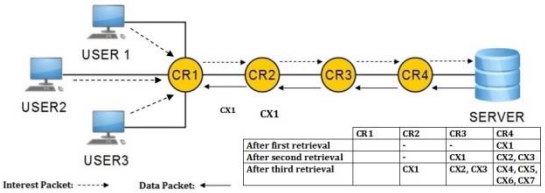

Figure 8: WAVE operations.

The sum of content chunks to be stored by a content-router is strongminded by the Chunk Marking Window (CMW), the CMW is determined by CMW state (n) and CMW base (x), Thus, CMW ranges will be from $\sum_{j=0}^{n-1} x^j + 1$ to $\sum_{j=0}^{n} x^j$. The WAVE operations are described in Figure 8.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.9 Cache-Aware Target Identification (CATT) [25].

The objectives of CATT are robustness, availability, diversity, and adaptability. A Data-packet is stored at a single content-router by CATT across the downloading path in every network. With the introduction of expected quality of content, CATT assists content-routers in forwarding Interest-packets. As shown in Figure 9, once a content-router stores a content, the expected quality of such router is denoted as Qc. Otherwise, the expected quality of the content at such router will be denoted as Qua(Ri).

$$\text{Qua(Ri)} = \frac{\text{Qc}}{\text{Hop}_r+1} \qquad (7)$$

The sum of hops from one router source to the base server as well as the content-router that caches the content is denoted as a Hop_r.

Instead of FIB, CATT employs a Potential Based Routing (PBR) for content retrieval. In CS, when there is no cached replica of the content demanded by Interest-packet, the content-router that receives the packet would examine the expected quality of content of its neighbors' PBR. Then, the content-router would select the neighbor with greatest expected quality of content to forward the Interest-packet.

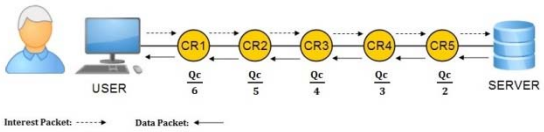

Figure 9: CATT operations.

4.10 A Chunk Caching Location and Searching Scheme (CLS) [26].

As an advanced version of MCD, CLS seeks to improve file download time and server workload. Hence, a maximum of one replica of a chunk is stored by CLS on the pathway amid a server and a requester. When there is a request, the copy of chunk will move one level down towards the requester. In the case of cache eviction, the chunk will move one level up towards the server. In CLS, a content-router will maintain a caching table that consists of several entries and each of the entries is a 4-tuple, namely the content-name (*Name*), the upstream content-router (*Content-routerIDin*), the downstream content-router (*Content-routerIDout*), and the number of hop to the content source (*Hop*).

As shown in Figure 10, in a caching trail, the value of Hop field would specify the caching position of the consistent content.

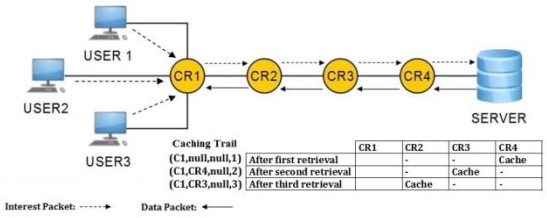

Figure 10: CLS operations.

When a content-router obtains an Interest-packet with content less in the CS, the content router can use the Hop number in the caching pathway to determine the name that is able to be employed to identify the subsequent hop of the Interest-packet, either the upstream or the downstream router.

The network administrator would determine the threshold value (Tha) in advance.

If Tha <= value of hop Then downstream else upstream.

4.11 Edge Caching [27].

As shown in Figure 11, edge caching allocates content nearer to its users by storing content in the latest content-router on a distribution pathway. The objective here is to decrease the number of hops needed to reach a content source. This in turn leads to the reductions of both time used for content delivery and traffic within a network caused by the transmission of content requests.

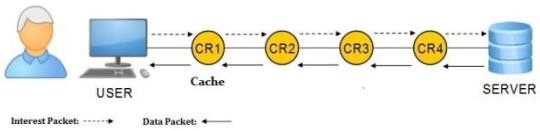

Figure 11: Edge Caching operations.

4.12 Distributed Cache Management (DCM) [28].

The features of *DCM* are the maximization of traffic volume served from the caches and the minimization of the bandwidth cost. According to DCM, a *Cache Manager* (CM) is installed so that its cache state is exchanged with other CMs. The decision about the replacement of cached content depends on the distributed on-line cache management algorithm employed by the CM.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Let Gn =<V,E>, where V is content-routers, and E is links between content-routers. For a better understanding, the main algorithm parameters are as below :

- S is the whole sum of saved content of an assumed network Gn.
- $r^s R_v$ is the inward demand ratio at Rv($Rv \in V$) based content s ($s \in S$).
- $hsRv$ is *true* once s is saved by Rv.
- hsRv= if (s is saved by Rv) Then 1 else 0.

DCM aims to obtain an applicable configuration in minimizing the cost of network traffic. A linear programming is formulated as below:

$$T(H) = \sum_{s=1}^{S} r^s R_v \quad dR_v R_m \qquad (8)$$

Where Rm represents the nearest cache-router that stores a replica of a given content s and dR_vR_m represents the number of hops between Ry and Rm.

The first constraint indicates that every a stored content is cached in at least one content-router. The additional restriction is the limited caching capacity for each content-router.

The myopic cache management algorithm is employed and explained as below:

- First stage: CM(Rv) chooses a content such as content s and determines if there is a locally cached content X.
- Second stage: If content s is yet to be cached by Rv, the traffic gain, gs = T(Hi) − T(H∗), will be estimated by CM(Rv) when it is saved by R_v later, which H* represents that hsRv is equal to true and the other cache formations persist constant. If content s has been cached, the traffic loss, ls = T(H∗) − T(Hi), will be estimated by CM(Rv) when Rv evicts the content s, where H* represents that hsRv is equal to untrue while supplementary cache formations persist constant.
- Third stage: CM(Rv) repeats the first and second steps until the cached contents are completely chosen. The highest existing gain content is regarded as a saved content candidate (for example i) while the content of a lowest traffic lost is considered a replacement applicant (for example j). CM(Rv) will fetch the content i and evict the stored content j when the relative gain, Rg = gi − lj, is positive. hiRn and hjRv are simultaneously set to be true and false, respectively.
- Fourth stage: The advertisement of new cache state is made by CM(Rv) to other CMs.
- Fifth stage: Repeat each step in other CMs until there is no extra beneficial replacement.

4.13 Popularity based Caching Strategy for Content Centric Networks (MPC) [29].

Instead of distributing widespread content on the way to the neighbors on a delivery path, the key focus of MPC is the distribution of popular content towards all neighboring content-routers. Under MPC, only popular content is cached. Each content-router in the network can store two types of information, namely “content name” and “popularity count”, hooked on a special table known as popularity table.

Figure 12 shows that every content-router across the downloading path sums the incoming needs for every content name locally and the data is stored in a popularity table.

If Content Popularity >= Threshold, Then Cache & Popular, Else Unpopular.

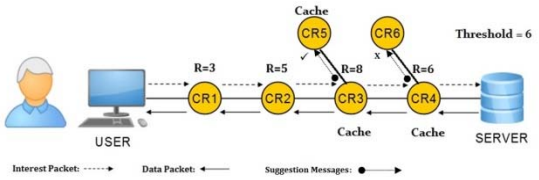

Figure 12: MPC operations.

Given a cached popular content, the particular content-router would send a message to its neighboring content-routers and recommend them to store the popular content. After that, the decision to cache the said content is made by the neighboring content-routers based on their respective caching policies.

4.14 Intra-AS Cache Cooperation [30].

By allowing the public visibility of cached content, Intra-AS Co can reduce the caching redundancy.

In Intra-AS Co, each content-router maintains two types of cache brief list, namely *Exchange Cache-Summary Table* (ECST) and *Local-Cache Summary Table* (LCST).

The ECST stores the information of cached contents at a router’s neighboring content-routers

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

while the LCST stores the information of the locally cached content chunks.

Furthermore, each content-router regularly advertises its summaries of local-cache to its directly linked neighbors. A content-router will store the information of router identifiers and content names into its ECST when summaries are received from its directly linked neighbors.

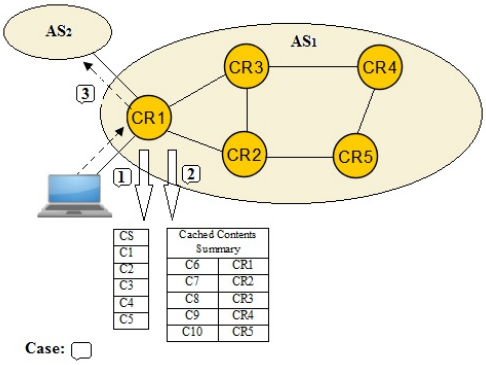

Figure 13: Intra-AS operations.

Once an Interest-packet reaches at a content-router, its cached copy will be searched in the router's CS. If there is no match found, the content-router will use its ECST to check on the availability of the desired content chunk cached in its directly linked neighbors. Based on the router's FIB, the Interest-packet will be promoted to the following hop when no cached replica is found, as shown in Figure 13.

4.15 Centrality-based Caching (CBC) [31].

The objectives of CBC are to reduce the latency of content delivery, alleviate traffic and congestion given that a cache hit will result in lesser paths traversed by a content, and reduce server load since one cache hit leads to one less request.

BC(R) signifies the betweenness centrality of each content-router.

$$BC(Ry) = \sum \frac{n(Ri, Rj, Ry)}{n(Ri, Rj)} \quad (9)$$

n(Ri , Rj ,Ry) : is the numeral of shortest paths from *Ri* to *Rj* through *Ry*. While, n(Ri,Rj) represents the number of shortest paths amid Ri and Rj.

As shown in Figure 14, Data-packet is stored at the "important" content-router with the greatest betweenness centrality (*BC*(R)) rate across the delivery path.

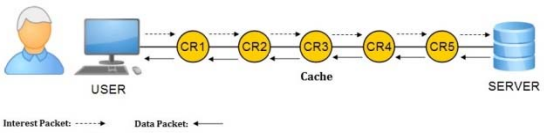

Figure 14: CBC operations.

For CBC, the header of Interest-packet includes a part that records the maximum *BC*(R) rate.

When an Interest-packet is promoted, the *BC*(R) values of the forwarding content-router and the one stored in the Interest-packet are associated. If the router's *BC*(R) rate is higher than the individual stored in the Interest-packet, thus the router will replace the *BC*(R) value of the Interest-packet with its own *BC*(R) value.

Similar to Interest-packet, the *BC*(R) values of the receiving content-router and the one stored in Data-packet are compared. If the values are the same, the content-router will cache the Data-packet into its CS.

4.16 Two Layers Cooperative Caching (TLCC) [32].

For TLCC, there are several groups of clustered CCN routers within one Autonomous System (AS). Using consistent hashing, each group collaborates in storing contents and eliminating duplicate contents.

One AS is made up of two groups, as shown in Figure 15, namely Upper-Layer Group (ULG) and Lower-Layer Group (LLG). ULG is linked to LLG or further AS, this connection will lead to the storage of chunks corresponding to the keys of respective router, whereas LLG is connected with users, this connection will lead to the storage of popular chunks based on keys and the popularity threshold value.

Two types of caching algorithms are employed in TLCC, namely Cache Decision for Upper Layer Group (CDULG) and Cache Decision for Lower Layer Group (CDLLG).

CDAULG: When ULG routers receive a chunk, the chunk name is checked and hashed to identify the corresponding router. ULG will cache the chunks if its hashed value is within the key range of current router. Otherwise, the chunk is then forwarded to the dedicated router.

CDALLG: When LLG routers receive the Data chunk, the routers examine the chunk name and Local Popularity Count (LPC). LLG will cache the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

chunks when there is an above-threshold LPC and the hashed value is within the key range of the current router. Otherwise, the current router will just forward the chunk to the client or other routers through requested face(s).

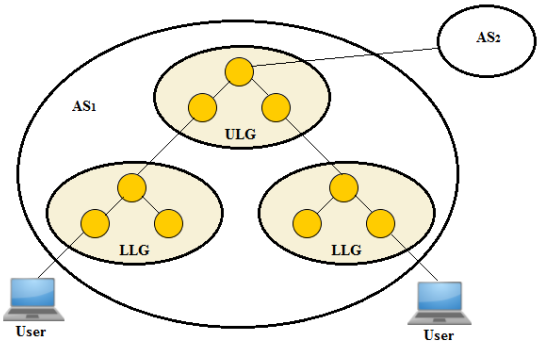

Figure 15: The groups of Intra-AS

4.17 A Distributed MAX-Gain In-network Caching Strategy in ICN [12].

The objectives of MAGIC are to reduce bandwidth consumption with a joint consideration of hop reduction and content popularity. Besides, cache replacement penalty is considered in the decision-making process for cache assignment to decrease the caching operations.

Let G =<V,E>, where V is content-routers, and E is associations between content-routers. For a better understanding, the main algorithm parameters are as below :

- M: The requested content where $M = \{m1, ...,m|M|\}$.
- r_v^m: The request rate (calculate as the sum of Interest-packet per second) at content-router v for content m.
- h_v^m: The number of hops from content-router v to the base server that preserve content m.
- Cv: The content saved at content-router v where $Cv \subset M$.
- ci: The cache capacity of content-router vi.

First, MAGIC calculates the potential place gain of cache content m at router v; second, it calculates the cache replacement penalty, then Local Gain of cache content m at router v.

$$\text{PlaceGain}_v^m = r_v^m \times h_v^m \quad (10)$$

$$\text{RePlacePenalty}_v = \begin{cases} \min m \in Cv \; r_v^m \times h_v^m, & \text{if } |Cv| = ci \\ 0 & \text{, if } |Cv| < ci \end{cases} \quad (11)$$

$$\text{LocalGain}_v^m = \text{PlaceGain}_v^m - \text{RePlacePenalty}_v \quad (12)$$

In MAGIC, if the drawback of substituting a saved content is greater than the gain of caching a new content, then there will be neither cached content eviction nor cache replacement.

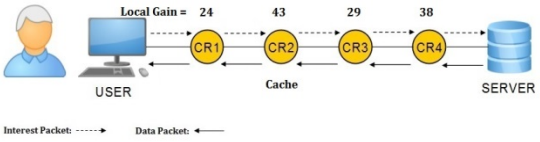

Figure 16: MAGIC operations.

In order to identify the content-router that has a greatest *LocalGain* for content *m,* an Interest-packet is attached with the maximum local cache-gain across the pathway in the latest denoted by *MaxGain* field. An Interest-packet with a zero *MaxGain* cost in the header is sent to request content *m.*

When the Interest-packet is received, the value of $LocalGain_v^m$ is estimated by each router *v* and the estimated cost is likened with the value contained in the *MaxGain* field.

When the recorded *MaxGain* value is lower than the local cache-gain of content-router *v,* the *MaxGain* cost in the Interest-packet will not be updated. When the Interest-packet arrives at a content-router or the initial server, the *MaxGain* part in the Interest-packet equals to the extreme local cache-gain across delivery path.

The Data-packet is attached with *MaxGain* value of the Interest-packet, and then the Data-packet will send to node that request and cache at content-router with the equivalent value as *MaxGain* value in the Data-packet. Figure 16 shows the fundamental MAGIC's operations.

4.18 In network Cashing for ICN with Partitioning and Hash-Routing (CPHR) [33].

The objective of CPHR is to fully utilize ICN's built-in caching capability. Hence, a new field known as *Egress Router* is included in each FIB entry of a content-router in CPHR. This is to keep track the movement of Interest-packet for the content name from a field to a content source. Also, an entrance router includes two extra parts in the header of Interest-packet.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- Cache-content Name - To document the position of cached content-chunk for the content-name in the Interest-packet.

- Egress-content Router - To show the name of outlet router conforming to the content-name in the Interest-packet.

Besides, content names are mapped to content-routers in CPHR via consistent hashing. Agreeing to the content-name hashed, the content-router will cache the content chunk for the content name hashed. Likewise, a content-router will hash the content name given by the Interest-packet and the Interest-packet is then forwarded to the content-router which the content-name is hashed to, as shown in Figure 17. This is done with the assistance of Cache-Name part in the Interest-packet.

The request will be satisfied when the content chunk is cached locally by its corresponding content-router. Otherwise, the Interest-packets will be forwarded to the content source with the assistance of the Outlet Router filed in the Interest-packet.

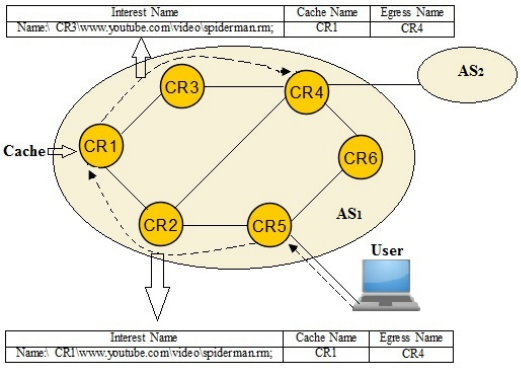

Figure 17: CPHR operations

4.19 Cooperative in Network Caching (CINC) [34].

CINC is designed as a supportive caching technique for huge audiovisual streams such as online TV services. Using a customary of organized CRs, CINC aims to reduce the number of inquiries for time shifted TV that are responded by servers beyond the Internet Service Provider network.

In CINC, q is the number of content-routers, *Lr* is label of a content-router, and *Sd* is the order number of a Data-packet.

If Lr = Sd mod q, then the strategy cache copy at content router, else forwards, to avoid caching redundancy.

To attain this objective, CINC includes dual new lists in every content-router, namely the Collaborative-Router Table (CRT) and the Collaborative-Content Store (CCS). CRT aims to document the information about reachability of other content-routers within a network, while CCS aims to document the content names cached by other content-routers.

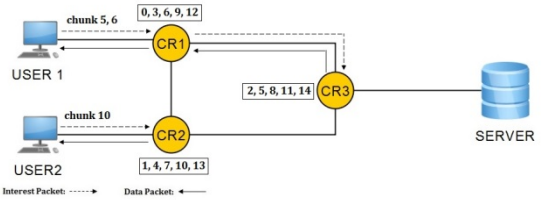

Figure 18: CINC operations.

4.20 Auction-based In-network Caching in ICN (BidCache) [13].

The key idea of BidCache is that the right of caching is determined via bidding process. The Interest message will then piggyback the winning bid value. By comparing the present bid cost in the Interest-message and the local-bid that can be issued by the cache, each caching content-router will determine their bidding decisions. Prior to the forwarding of Interest message to the next-hop, the present uppermost bid value in the Interest-message is replaced with the bid cost of the content-router that decides to bid.

The information about the winner in the auction will be distributed to the participating caching content-routers using piggy-backing in the return Data-message.

1) Issuing a bid

The bid value contains a weighted (*w1 – wk*) sum of local and non-local factors that are comparative to the requested content. Each parameter has distinct weight and this weight would influence the caching probability at a specific content-router. For example, a greater weight in a central content-router would lead to higher probability to cache popular content.

The weights can be influenced by the nature of content-router, network features and practice, or the adjustment made by the network manager. An example of bid construction is shown below:

$$BID = < local\ factors > + < context - request >$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Motivating local factors to study for the bid cost could contain:

•The value of Least Recently Used (LRU)
• The value of Least Frequently Used (LFU)
• Link Bandwidth (BW)
• Cache size (CS)
• Hop Count (HOP)
• Link Delay (LD)

The bid cost is computed as follows:

$$BID = w1 * f1(LRU) + w2 * f2(LFU) + w3 * f3(BW) + w4 * f4(CS) + w5 * f5(HOP) + w6 * f6(LD) \quad (13)$$

Functions *f1*, *f2*, etc. allow the normalization of each parameter value for all caching content-routers. This is to ensure that the bids from content-routers with possibly huge dissimilar sceneries or abilities are adjusted with a similar scale.

On the other hand, the weights *w1*, *w2*, etc. allocate comparative importance to definite factors which are highly-prioritized based on the opinion of content-router. For example, a large weight is assigned to the certain content-routers with small CS to ensure the CS "dominates" the last bid cost.

Algorithm 1 demonstrates the process after a bid is issued by a caching content-router. Initially, an Interest-message is sent with r.bid_winset to an indeterminate value. Then, the cache might place a bid if its estimated bid is equivalent to the bid transported in the Interest-message and this may lead to the previous bid value of the cache be overridden. Otherwise, the content-router will not bid.

Algorithm 1: Allotting a bid

```
r = receive(Interest)
c = r.request_context
l = local_parameters
bid = issueBid(l,c)
if bid >r.bid_win or r. bid_win = = "undefined" then:
r.bid_win = bid
bid_issued = True
else:
bid_issued = False
r.request_context = calculateRequestContext()
forward(r)
```

2) Deciding the auction winner

When the producer or the holder of the duplicated requested data item receives the Interest message, the data item is returned across the Interest path. Moreover, the Data message will carry the information concerning the auction winner. As demonstrated by Algorithm 2, each content-router on the path will use this information to determine if it has the winning bid.

In the algorithm, an assumption is made where the Data message sent by the producer or the holder of duplicated requested data-item set the *r.can_cache* part as True. When the content-router with the winning bid obtains the Data- message, the data-item is stored and the *r.can_cache* part is set as non-true. This is to inform the following content-router that the bid has been won by previous content-router, thus the item does not need to be cached.

Algorithm 2: Determining the auction victor

```
r = receive(Data)
can_cache = r.can_cache
if can_cache = = True then:
if bid_issued = = True:
cache(Data)
r.can_cache = False
forward(r)
```

Figure 18 shows the process of an auction.

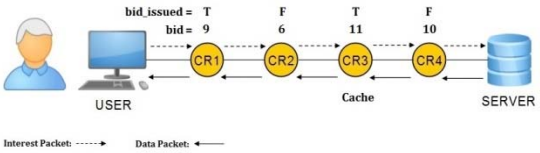

Figure 19: BidCache operations.

4.21 Link Congestion and Lifetime Based In-Networking Caching Schemes (LCLCS) [14].

With the consideration of betweenness centrality, link congestion, and content popularity, LCLCS aims to sustain acceptable cache rate and significantly optimize network delay

The process of LCLCS begins with an network analysis that leads to the identification of "important" content-routers in the content delivery path. After that, LCLCS applies the calculated lifetime to decide the eviction of current cached content. This is to ensure sufficient space for the incoming content to be cached.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

In the centrality of content-routers within a network, the betweenness centrality is the key metric as shown below:

$$C_B(v) = \sum_{i \neq v \neq j \in V} \frac{S_{i,j}(v)}{S_{i,j}} \quad (14)$$

Where $S_{i,j}(v)$ is the sum of pathways in the range of i to j over content-router v, and $S_{i,j}$ is the sum of pathways from i to j. Using the central management system with an assumption of known content-router's betweenness value, the delivery route can be estimated offline or online.

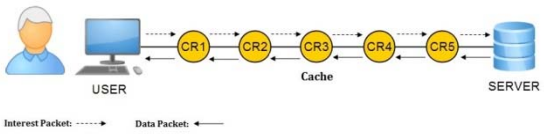

Figure 20: LCLCS operations.

The available bandwidth of link e is denoted as $B_i = C_i/N_i$, where N_i represents the sum of flows fleeting through the connection and C_i represents the capacity of the link.

Delay saving, $ST_{r,v}$ refers to the time difference between obtaining content from the initial server and from the content-router v (local-cache). While L_D refers to a set of connections bridging the cache to the downstream requesters, L_U refers to a set of connections bridging the upstream source to the cache. For every connection 1 of L_D and L_U, the set of available link capacities in L_D is defined as B_{LD} and the set of available link capacities in L_U is defined as B_{LU}. Given that greatest bottleneck connections decide the entire throughput of the data flow, the throughput from the cache to the requester is the min (B_{LD}) and the throughput of the both ends in the pathway is the min ($B_{LD} \cup B_{LU}$). Then, the approximation of postponement saving via content saving on content-router is shown as below:

$$ST_{r,v} = \frac{S}{\min(B_{LU} \cup B_{LD})} - \frac{S}{\min(B_{LD})} \quad (15)$$

The content size is denoted as S. High $ST_{r,v}$ value indicates longer duration is needed for contents to be cached in local. The purpose of this is to decrease the content download suspension.

Popularity shows the probability of content requested over a specific future timeframe. Given R contents with associated popularity, the weights of popularity are estimated as below:

$$\text{weight}_r = \frac{P_r}{\sum_{r=1}^{R} P_r} \quad (16)$$

As the fundamental concept of LCLCS, lifetime determines the content storage and content deletion. Each content has a lifetime and its replica obtains the content's lifetime before being included to the cache. When the content-router wants to cache a content but there is insufficient location, the content of a smallest lifetime is eliminated. According to this information, the lifetime of cache can be represented as below:

$$Lifetime_r = \frac{C_1(ST_{r,v} * \text{weight}_r)^{C_2}}{Size} * base\_time \quad (17)$$

Given constant C_1 and C_2, size is content size, and base_time is the basal cache time. The reason of including the content size is to enhance cache hit rate by caching compact object instead of the huge one. Given the dynamic nature of network state and content popularity, the concept of lifetime with diminishing nature is selected to ensure the feasible eviction of content that is difficult to obtain or popular from a cache.

When a content request is initiated, the relatively "important" betweenness set is documented by the request packet across the distribution path. The header of request packet will include the set and the field of sets is given as [base_set, max_set]. When the content distribution begins, the initial magnitude of set is given as base_set. For each three hops, there will be a unit increase in the size of set, equal to the greatest value of max_set. The size of set will decide the number of "important" content-routers. Prior to cache hit, the betweenness value of all intermediate content-routers will be compared and replaced with the betweenness value of C_{BN} and the set will be constantly updated across the delivery path. Hence, C_{BN} will carry the most important betweenness of content-routers across the path. While the data are transmitted at the hit content-router, the value of C_{BN} is reproduced in the Data-packet. En route to the requester, the betweenness of each content-router is compared with the attached C_{BN}. Then, caching will happen if there is a match and sufficient space for the object. If there is a match but space is insufficient, expired contents will be removed so that content-router can cache the new object. However, the Data-packet will be promoted to the following hop when no match or no expired contents when there is insufficient space.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.22 Adaptive Prioritized Probabilistic Caching Algorithm for Content Centric Networks [15].

Using caching probability as a foundation, the adaptive prioritized probabilistic caching algorithm aims to provide solutions for the limitations of previous caching algorithms. The technique emphasizes on content popularity and multiple priorities of Data-packets. Nonetheless, the caching probability considers the priority of data layers in its estimation.

Since certain Data-packets are more important, the priorities of Data-packets are divided into layers. The levels of importance for data layers are quantified and shown in Table 1.

Table 1: Data-Layers significance values

Layers	Base-Layer (i=0)	Improved-Layer (1)	Improved-Layer (2)	Improved-Layer (3)
γ i	1.0	0.3	0.2	0.1

A random real number between [0,1] is generated when data arrive at CCN router. The generated number is then compared with the caching probability. If the caching probability is greater than the generated number, the Data-packet will not be saved in the CS, as shown in Figure 20. The types of calculation component include factors and events. The factors include additional factor (ac), caching probability (P), priority values (γ), and multiplicative factor (mc). According to events such as cache-miss and cache-hit, CCN router can adjust the caching probability in its estimation process. Cache-miss refers to a situation where the incoming Interest-packets are not fulfilled by the cached Data-packets. Since the requesters do not need the current contents in the CS, CCN router will cache the subsequent incoming high probability content. On the other hand, cache-hit refers to the situation where the incoming Interest-packets are satisfied by the cached Data-packets. With current high cache-hit, CCN router will decrease the probability of subsequent received Interest-packets. Since data are divided into multiple layers, each layer has its own caching probability. Given n layers of data, CCN-router will estimate and save caching probabilities of a value of n. Eq. (18) estimates iP at time t γ

$$P_i(\mathrm{t}+\mathrm{S}) = \begin{cases} 1 \quad , P_i(\mathrm{t}) \; is \; greater \; than \; 1 \\ P_i(\mathrm{t}) * (c_{\mathrm{m}} * \gamma_i), \text{hit is occured} \\ P_i(\mathrm{t}) + (c_a * \gamma_i), \text{Elsewhere} \end{cases} \quad (18)$$

where cm, ac ∈ (0, 1) .

As shown in Eq. (18), the caching probability at time t+S is estimated based on the preceding caching probability at time t. Once an Interest-packet reaches at CCN router, its consistent Data-packet is searched in the CS and the caching probability will be updated contingent upon a cache event. In the case of cache-miss, the multiplication of the supplementary issue and the priority value of data-layer is added into the caching probability. In the case of cache-hit, the product of multiplication factor and priority value is multiplied with caching probability and this leads to lower cache probability.

The decision of caching Data-packets at CCN routers is assessed depending the importance of the received Data-packet and the caching likelihood of received Data-packet layer. For instance, given that the inward Data-packet at time t+S is the base-layer, the caching decision is made through comparison of random real number and caching probability $P_0(\mathrm{t}+\mathrm{S})$.

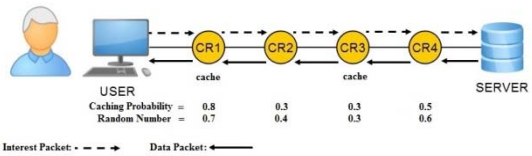

Figure 21: APP operations

4.23 Popularity and gain based caching scheme (PGBCS) [16].

The existing caching strategies lack choices in content placement and balanced distribution, thus, which lead to the problem of high user access time delay and low cache hit ratio. To solve the problem, PGBCS is proposed. The used symbols in PGBCS are defined in table 2.

Table 2: Symbols for PGBCS

Notation	Meaning
o	Content chunk
vi	The identify of i content router
w_i	The average value of content chunks cached in vi
S_i	The occupied space of vi
r_i	The space of *vi* expected to take up
Fi	The free space of vi
C_i	The overall cache space of vi
di	The distance between *vi* and user
D	The total length of Interest path
qi	The ratio of *di* and D
M (vi)	Caching gain of vi

There are four steps to calculate caching gain as following:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Step 1: calculating the average value of the content chunks cached in vi.

$$w_i = \frac{1}{n}\sum_{j=1}^{n}(O_j) \quad (19)$$

Step 2: Computing the utilization rate of vi.

$$p_i = \frac{S_i + r_i}{c_i} \quad (20)$$

Step 3: The distance factor.

$$q_i = \frac{d_i}{D} \quad (21)$$

Step 4: The calculation formula of vi caching gain index.

$$M(v_i) = \frac{1}{w_i} \cdot \frac{1}{p_i} \cdot \frac{1}{q_i} \quad (22)$$

When interest comes to the content router, check if the content router has free space or not, then it calculates the caching gain. When the cache hit, If content routers have free space on the download path, PGBCS will cache the chunk on the content router that has free space and is closest to the user as shown in Figure 22, if not, the strategy will cache the content on the content router with maximum caching gain as shown in Figure 23. When the content arrives to cache at the content router, PGBCS checks if the content chunk popularity value is not less than the lowest content chunk popularity value in the content router, the strategy will evict the content chunk that has the lowest value to cache the new content chunk, if no, the content chunk will be discarded.

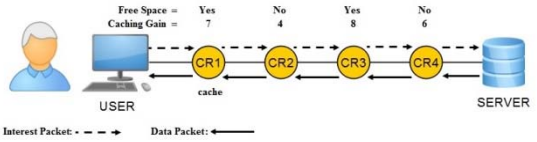

Figure 22: PGBCS operations (Case 1)

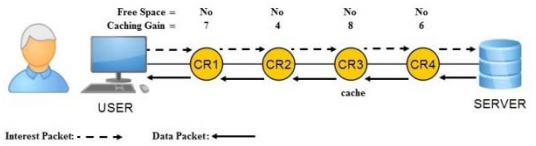

Figure 23: PGBCS operations (Case 2)

5. SUMMARY OF CATEGORIES OF CACHING MECHANISMS

After introducing the caching strategies and define the categorizations of caching strategies, in this lesson, the caching strategies will be classified based on these categorizations to realize and understand what are the similarities and differences between the principles of caching strategies to take a decision to cache the content.

In Path caching, the strategies are categorized into On-Path and Off-Path. The On-Path is used to clarify if the caching strategy caches a copy or copies of the content along downloading path between the server and requester such as LCE, LCD, MCD, Prob(p), Breadcrumbs, Probcache, Probcache+, WAVE, CATT, CLS, MPC, Intra-AS, CBC, TLCC, MAGIC, CPHR, Edge, BidCache, PGBCS, APP, and LCLCS. The Off-Path is used to clarify if the caching strategy caches a copy or copies of the content out the downloading path between the server and requester such as CINC and DCM.

In redundancy caching, the caching strategies are categorized into redundancy and non-redundancy caching. The redundancy caching is used to clarify if the caching strategy caches more than one copy of the content along downloading path between the server and requester such as LCE, LCD, Breadcrumbs, and Intra-AS. while the non-redundancy caching is used to clarify if the caching strategy caches one copy of the content along downloading path between the server and requester such as MCD, Prob(p), Probcache, Probcache+, WAVE, CATT, CLS, MPC, DCM, CBC, TLCC, MAGIC, CPHR, CINC, Edge, BidCache, PGBCS, APP, and LCLCS.

In popularity caching, the caching strategies are categorized into popularity and non-popularity caching. The popularity caching is used to clarify if the caching strategy based on the number of requests for content to select a content router to cache the content, such as LCD, MCD, WAVE, CLS, MPC, DCM, MAGIC, PGBCS, and LCLCS. Otherwise, the caching strategy considers as non-popularity caching such as LCE, Prob(p), Breadcrumbs, Probcache, Probcache+, CATT, Intra-AS, CBC, TLCC, CPHR, CINC, Edge, BidCache, and APP.

Generally, the Cooperative caching is used to clarify if the caching strategy caches the content based on information from other content routers. In Cooperative caching, the caching strategies are categorized into Implicit Cooperation Caching, explicit Cooperation Caching and non-Cooperation Caching. If the content router advertises information about what it has contents, in this case, the caching strategy consider as explicit Cooperation Caching, such as CATT, DCM, Intra-AS, TLCC, CPHR, and CINC. while if the content router advertises other information such as the number of requests for content or number of hops from the requester, in this case, the caching strategy considers as Implicit

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Cooperation Caching such as LCD, MCD, Breadcrumbs, Probcache, Probcache+, WAVE, CLS, MPC, CBC, MAGIC, Edge, BidCache, PGBCS, and LCLCS. Otherwise, the caching strategy considers as non-Cooperation Caching such as LCE, Prob(p), and APP.

In probability caching, the strategies are categorized into probability and non-probability caching. The probability caching is used to clarify if the caching strategy caches the content based on probability such as Prob(p), Probcache, Probcache+, and APP. Otherwise, the caching strategy considers as non- probability caching such as LCE, LCD, MCD, Breadcrumbs, WAVE, CATT, CLS, MPC, DCM, Intra-AS, CBC, TLCC, MAGIC, CPHR, CINC, Edge, BidCache, PGBCS, and LCLCS.

Generally, the distribution caching is used to clarify if the caching strategy selects content router to cache the content based on content routers position such as the distance between content router and server, the distance between the content router and requester, edge router, neighbor router, and center router. In the distribution caching, the caching strategies are categorized into distribution Caching, and non-distribution Caching. According to that, LCD, MCD, Probcache, Probcache+, WAVE, CATT, CLS, DCM, Intra-AS, CBC, TLCC, MAGIC, CPHR, CINC, Edge, BidCache, PGBCS, and LCLCS consider as distribution Caching. While LCE, Prob(p), Breadcrumbs, MPC, and APP consider as non-distribution Caching.

In content-type caching, the strategies are categorized into chunk caching and content caching. The chunk caching means the caching strategy divides the content into chunks, and each chunk needs interest message such as Probcache, Probcache+, WAVE, CLS, MPC, Intra-AS, TLCC, CINC, PGBCS, and APP. Otherwise, the caching strategy considers as content caching such as LCE, LCD, MCD, Prob(p), Breadcrumbs, CATT, DCM, CBC, MAGIC, CPHR, Edge, BidCache, and LCLCS.

The review of the latest caching strategies and its categorizations is summarized in *Table 3*.

Table 3: Categories Of Caching Mechanisms

Category	Caching Mechanisms
Implicit Cooperation Caching	LCD, MCD, Breadcrumbs, Probcache, Probcache+, WAVE, CLS, MPC, CBC, MAGIC, Edge, BidCache, PGBCS, LCLCS.
explicit Cooperation Caching	CATT, DCM, Intra-AS, TLCC, CPHR, CINC.
Non-cooperation Caching	LCE, Prob(p) , APP.
On-path Caching	LCE, LCD, MCD, Prob(p), Breadcrumbs, Probcache, Probcache+, WAVE, CATT, CLS, MPC, Intra-AS, CBC, TLCC, MAGIC, CPHR, Edge, BidCache, PGBCS, APP, LCLCS.
Off-path Caching	CINC, DCM
Popularity Caching	LCD, MCD, WAVE, CLS, MPC, DCM, MAGIC, PGBCS, LCLCS.
Non-Popularity Caching	LCE, Prob(p), Breadcrumbs, Probcache, Probcache+, CATT, Intra-AS, CBC, TLCC, CPHR, CINC, Edge, BidCache, APP.
Distribution Caching	LCD, MCD, Probcache, Probcache+, WAVE, CATT, CLS, DCM, Intra-AS, CBC, TLCC, MAGIC, CPHR, CINC, Edge, BidCache, PGBCS, LCLCS.
Non-Distribution Caching	LCE, Prob(p), Breadcrumbs, MPC, , APP.
Probability Caching	Prob(p), Probcache, Probcache+, APP.
Non-Probability Caching	LCE, LCD, MCD, Breadcrumbs, WAVE, CATT, CLS, MPC, DCM, Intra-AS, CBC, TLCC, MAGIC, CPHR, CINC, Edge, BidCache, PGBCS, LCLCS.
Redundancy Caching	LCE, LCD, Breadcrumbs, Intra-AS.
Non-Redundancy Caching	MCD, Prob(p), Probcache, Probcache+, WAVE, CATT, CLS, MPC, DCM, CBC, TLCC, MAGIC, CPHR, CINC, Edge, BidCache, PGBCS, APP, LCLCS.
Chunk Caching	Probcache, Probcache+, WAVE, CLS, MPC, Intra-AS, TLCC, CINC, PGBCS, APP.
Content Caching	LCE, LCD, MCD, Prob(p), Breadcrumbs, CATT, DCM, CBC, MAGIC, CPHR, Edge, BidCache, LCLCS.

6. CONCLUSION

In general, it is clear that caching strategies have a significant impact on the performance of the In-network cache, which manages the distribution of contents in different regions to make it easy to retrieve on demand.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

In this review, the importance of the In-network cache has been clarified, in addition to clarifying the importance of the caching strategy that has a significant role in improving performance, such as reducing server load, congestion, and delay caused by users. Furthermore, caching strategies have been discussed and described clearly in terms of the principle and the purpose.

In this review, the categorizations of caching strategies are defied, then, the caching strategies are classified based on these categorizations that help the reader to understand the principle of caching strategy, and recognize the similarities and differences between these strategies.

REFERENCES

[1] M. A. Naeem and S. A. Nor, "A survey of content placement strategies for content-centric networking," *AIP Conf. Proc.*, vol. 1761, 2016.

[2] A. A. Barakabitze and T. Xiaoheng, "International Journal of Advanced Research in Computer Science and Software Engineering Caching and Data Routing In Information Centric Networking (ICN): The Future Internet Perspective," vol. 4, no. 11, pp. 8–20, 2014.

[3] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent Advances in Information-Centric Networking based Internet of Things (ICN-IoT)," vol. 14, no. 8, pp. 1–31, 2018.

[4] I. Abdullahi, S. Arif, and S. Hassan, "Survey on caching approaches in Information Centric Networking," *J. Netw. Comput. Appl.*, vol. 56, no. 2015, pp. 48–59, 2015.

[5] D. Saxena and I. I. T. Roorkee, "Named Data Networking: A Survey," *Comput. Sci. Rev. Elsevier*, vol. 19, pp. 15--55, 2016.

[6] "Cicn." [Online]. Available: https://wiki.fd.io/view/Cicn. [Accessed: 29-Jan-2018].

[7] T. Koponen *et al.*, "A data-oriented (and beyond) network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, p. 181, 2007.

[8] C. Bernardini, T. Silverston, and A. Vasilakos, "Caching Strategies for Information Centric Networking : Opportunities and Challenges," pp. 1–14, Jun. 2016.

[9] N. El, H. Ben, Y. Barouni, J. Ben Hadj, and K. Ben Driss, "Performance Analysis of In-Network Caching in Content-Centric Advanced Metering Infrastructure," vol. 7, no. 11, pp. 108–115, 2016.

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

[11] A. Gupta, S. Shailendra, and A. Girish, "Analysis of In-Network Caching for ICN," 2016.

[12] J. Ren *et al.*, "MAGIC: A distributed MAx-Gain In-network Caching strategy in information-centric networks," *Proc. - IEEE INFOCOM*, pp. 470–475, 2014.

[13] A. S. Gill, L. D'Acunto, K. Trichias, and R. Van Brandenburg, "BidCache: Auction-based in-network caching in ICN," *2016 IEEE Globecom Work. GC Wkshps 2016 - Proc.*, pp. 0–5, 2016.

[14] J. Kong, L. Rui, H. Huang, and X. Wang, "Link congestion and lifetime based in-network caching scheme in Information Centric Networking," *IEEE CITS 2017 - 2017 Int. Conf. Comput. Inf. Telecommun. Syst.*, pp. 73–77, 2017.

[15] W. Sirichotedumrong, W. Kumwilaisak, S. Tarnoi, and N. Thatphitthukkul, "Adaptive prioritized probabilistic caching algorithm for content centric networks," *Eng. J.*, vol. 21, no. 6 Special Issue, pp. 11–22, 2017.

[16] Z. Fan, Q. Wu, M. Zhang, and R. Zheng, "Popularity and gain based caching scheme for information-centric networks," vol. 7, no. 30, 2017.

[17] A. Kalla and S. K. Sharma, "A constructive review of in-network caching: A core functionality of ICN," *Proceeding - IEEE Int. Conf. Comput. Commun. Autom. ICCCA 2016*, pp. 567–574, 2016.

[18] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Information-Centric Networking based Caching and Naming Schemes for Internet of Things: A Survey and Future Research Directions," no. October, 2017.

[19] M. Meddeb *et al.*, "How to cache in ICN-based IoT environments ? To cite this version : HAL Id : hal-01575386 How to cache in ICN-based IoT environments ?," 2017.

[20] A. Ioannou and S. Weber, "Towards on-path caching alternatives in Information-Centric Networks," *Proc. - Conf. Local Comput. Networks, LCN*, pp. 362–365, 2014.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[21] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Perform. Eval.*, vol. 63, no. 7, pp. 609–634, 2006.

[22] E. J. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, best-effort content location in cache networks," *Proc. - IEEE INFOCOM*, pp. 2631–2635, 2009.

[23] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," *ICN'12 - ACM Proc. Information-Centric Netw. Work.*, pp. 55–60, 2012.

[24] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," *Proc. - IEEE INFOCOM*, pp. 316–321, 2012.

[25] S. Eum, K. Nakauchi, Y. Shoji, N. Nishinaga, and M. Murata, "CATT: Cache aware target identification for ICN," *IEEE Commun. Mag.*, vol. 50, no. 12, pp. 60–67, 2012.

[26] Y. Li, T. Lin, H. Tang, and P. Sun, "A chunk caching location and searching scheme in Content Centric Networking," *IEEE Int. Conf. Commun.*, pp. 2655–2659, 2012.

[27] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, and A. Ghodsi, "Less Pain , Most of the Gain : Incrementally Deployable ICN," pp. 147–158, 2013.

[28] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed cache management in information-centric networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 10, no. 3, pp. 286–299, 2013.

[29] C. Bernardini, T. Silverston, and O. Festor, "MPC : Popularity-based Caching Strategy for Content Centric Networks," pp. 3619–3623, 2013.

[30] J. M. Wang, J. Zhang, and B. Bensaou, "Intra-AS cooperative caching for content-centric networks," *Proc. 3rd ACM SIGCOMM Work. Information-centric Netw. - ICN '13*, p. 61, 2013.

[31] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache 'less for more' in information-centric networks (extended version)," *Comput. Commun.*, vol. 36, no. 7, pp. 758–770, 2013.

[32] K. Thar, S. Ullah, and C. S. Hong, "Two layers cooperative caching in Content Centric Networks," 2014.

[33] S. Wang, J. Bi, J. Wu, and A. V. Vasilakos, "CPHR: In-network caching for Information-Centric Networking with Partitioning and Hash-Routing," *IEEE/ACM Trans. Netw.*, vol. PP, no. 99, pp. 2742–2755, 2015.

[34] Z. Li, G. Simon, Z. Li, and G. Simon, "for cooperative in-network caching Time-Shifted TV in Content Centric Networks : the Case for Cooperative In-Network Caching," 2011.