American Journal of Interdisciplinary Research and Development
ISSN Online: 2771-8948
Website: www.ajird.journalspark.org
Volume 20, Sep., 2023

ENERGY EFFICIENT TECHNIQUE AND ALGORITHM BASED ON ARTIFICIAL INTELLIGENCE IN CONTENT DELIVERY NETWORKS

Bakhtiyor Makhkamov,
Nazirjon Khasanov
Tashkent University of Information Technologies
Named after Muhammad al-Khwarizmi

Abstract:
Due to the large number of servers and network infrastructure required to deliver content to users, content delivery networks (CDN) consume a large amount of energy. CDN use several strategies to reduce energy consumption, such as server consolidation, dynamic provisioning, and load balancing. However, these strategies do not take into account the popularity of the content being presented. Therefore, a mechanism to improve energy efficiency based on content popularity has been developed in CDN. The main function of the mechanism is to make maximum use of the cache servers' memory capacity at the expense of optimal service to user requests and to increase the service performance of cache servers to user requests. To achieve this, based on machine learning algorithms using user requests and attributes of video files, predicting the probability of video content becoming popular and storing videos with the highest popularity index on edge cache servers is caught.

Keywords: Content delivery network (CDN), cache, content, Artificial intelligence, Machine learning, predicting, optimizing.

Introduction

Cache Placement and Cache Offloading processes are important in CDN to ensure the speed and reliability of content delivery to users around the world. Cache deployment is the distribution of CDN cache across geographic regions to increase content availability and reach for users. Cache swapping is the process of transferring cached data from a central CDN server to servers closer to end users. These are detailed below.

Caching technologies are used in content delivery networks to distribute video services in network segments that are maximally close to users. Since the capacity of caches is limited compared to the capacity of servers, they need to store the most requested content. Caching highly requested content significantly reduces access latencies, server overload, the amount of data to be cached, and resource costs required for caching [1].

Figure 1 shows the steps involved in delivering content by caching video services. In the typical case of video services, a user makes a video request to a server, and if the requested video is available in the cache serving the user, the user retrieves the video from that cache. If the video is not available in the cache, then the user's video request is forwarded to the remote server. Here, the first case is called caching, and the second case is called non-caching. The cache server updates its caches by replacing existing cached videos with newly requested videos, which is controlled by the cache update policy.

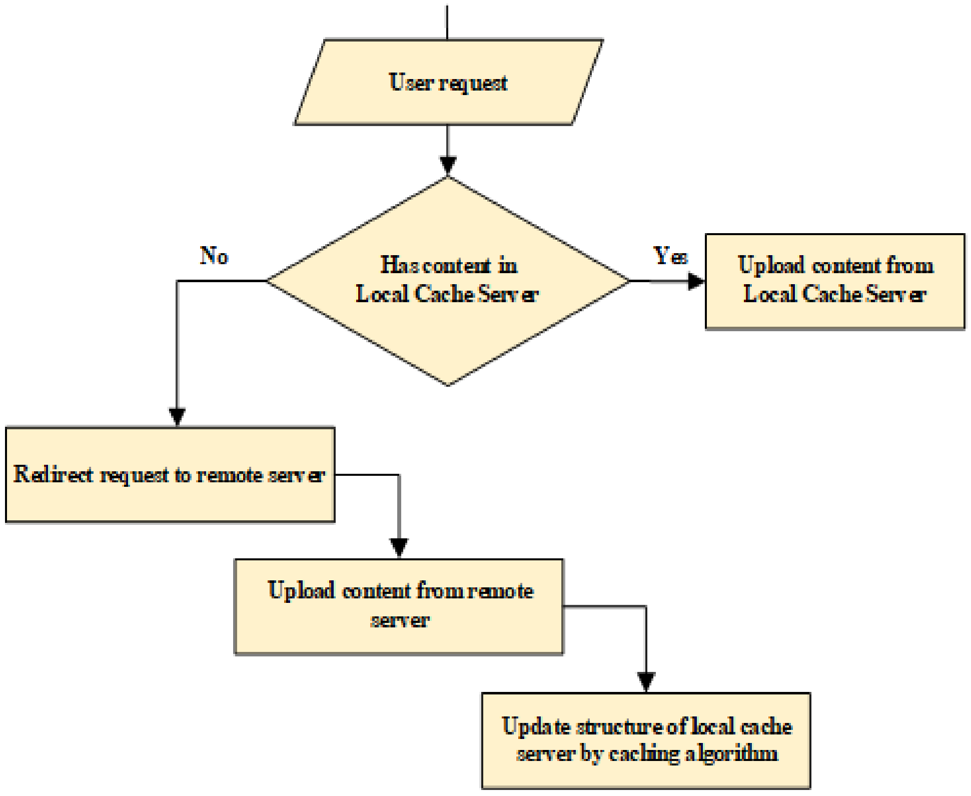
User request
Has content in Local Cache Server
No
Yes
Upload content from Local Cache Server
Redirect request to remote server
Upload content from remote server
Update structure of local cache server by caching algorithm

Figure 1. Content delivery process

The content delivery process described above has a simple structure and is appropriately designed for simple cache servers. In the event that the content of the provided service is not available in the local cache server, the user's request is directed directly to the main server, and in the process of making such a request, the resources of the network and other related elements costs. In the case of caching with a complex structure, the process of caching the content of the provided service is carried out hierarchically, which maximizes the caching priorities.

Caching has been widely used in content-based networks. Content placement algorithms are used to determine the order of routers from the actual content source to the user. Operational cache placement algorithms are important for Internet Service Providers (ISP) to reduce traffic and achieve good CDN performance. LCE (Leave Copy Everywhere), LCD (Leave Copy Down) and Prob algorithms are the last caches to determine the best in terms of lead time.

Materials and methods

Energy efficient VoD in content delivery network. Caching content locally results in a shorter route to content and lower power consumption as a result. However, this strategy leads to increased device power consumption by deploying local caches. Therefore, the optimal cache size is a function of the above two metrics, and the optimal value between them should be found. This evaluation aims to minimize the power consumption of the video service by optimizing the size of the caches deployed in the nodes. This considers a Video-on-Demand (VoD) service combining IP over a WDM network with the network architecture illustrated in Figure 2.

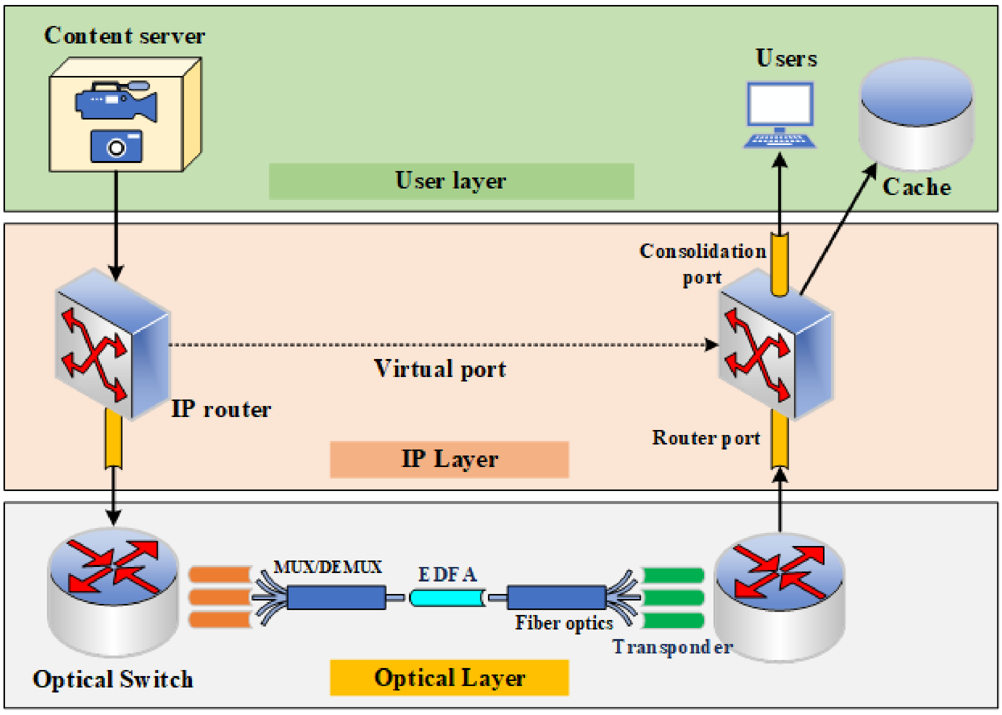

Figure 2. WDM-based content delivery architecture

VoD is an energy-intensive service due to the high energy consumption associated with the storage and delivery of large multimedia files. Keeping the most popular content close to the network boundary is one effective way to reduce the energy consumption of video services. We have estimated the energy consumption of video over IP in a WDM network. It explores the power savings introduced by using caches in the core nodes. Using a linear programming model, it is designed to optimize the size of the cache at each node in the network in order to minimize power consumption. The linear programming model is extended to account for variable caches, meaning that caches equipped with sleep capabilities allow inactive parts to be turned off when traffic is low. The model finds the optimal cache size for each node for different times of the day, which allows to achieve the best energy efficiency [2-3].

A variable-cache linear programming model finds the optimal cache size of each node that varies throughout the day. In the model, caches are assumed to have sleep capabilities, so inactive portions of the cache can be put to sleep. The goal is to explore potential additional power savings while using fixed caches and analyze changes in the optimal cache size that minimizes power consumption as network traffic changes.

This linear programming model finds the optimal cache size to be placed at each node in the network to minimize power consumption. Caches are assumed to have full working memory throughout the day.

The router ports required for each band wavelength at time t are determined by expression (1):

$$\sum_{i \in N} P_p \left(AP_{it} + \sum_{j \in Nm_i : i \neq j} C_{ijt} \right) \qquad (1)$$

where: AP_{it} – aggregation ports on node i at time t; C_{xyt} – the wavelengths in the virtual connection from node x to node y at time t.

The power consumption of the optical switch at time t is determined by expression (2):

$$\sum_{i \in N} Po_{it} \qquad (2)$$

where: Po_{it} – the power consumption of optical switch i at time t.

transponders at time t:

$$\sum_{i \in N} \sum_{i \in Nm_i} Pt \square w_{ijt} \qquad (3)$$

amplifiers at time t:

$$\sum_{i \in N} \sum_{i \in Nm_i} Pa \square Amp_{ijt} f_{ijt} \qquad (4)$$

Multiplexers/demultiplexers at time t:

$$\sum_{i \in N} \sum_{i \in Nm_i} Pmd\, f_{ij} \qquad (5)$$

Caches installed at time t:

$$\sum_{i \in N} \varnothing M \qquad (6)$$

It should be noted that the model does not assume a simple symmetric case, and therefore the number of lightpaths from node i to node j may differ from the number of lightpaths in the opposite direction. Basically, f_{ij}, w_{ijt}, and C_{ijt} are not necessarily equal to f_{ji}, w_{jit}, and C_{jit}, respectively. In this case, uplink traffic is video traffic uploaded from nodes to video servers, downlink traffic is video traffic downloaded from video servers to nodes, and regular traffic is traffic that cannot be cached (e-mail, live video, dynamic content, etc.)

The objective of the proposed integrated linear programming model is to minimize the daily power consumption of the network while overcoming a number of current and power constraints. A complete linear programming model is defined as:

$$\sum_{t\in T}\left(\begin{array}{l}\sum_{i\in N}P_{p}\left(AP_{it}+\sum_{j\in Nm_{i};i\neq j}C_{ijt}\right)+\sum_{i\in N}Po_{it}+\sum_{i\in N}\sum_{j\in Nm_{i}}Pt\square W_{ijt}+\\ \sum_{i\in N}\sum_{j\in Nm_{i}}Pa\square Amp_{ij}f_{ij}+\sum_{i\in N}\sum_{j\in Nm_{i}}Pmd\square f_{ij}+\sum_{i\in N}\varnothing M_{it}\end{array}\right) \qquad (7)$$

Through expression (7), the power consumption of the network can be calculated by summing the power consumption of different network components at each time point. The calculation of the optimal value of this analytical model is carried out based on the mechanism presented in Figure 3.

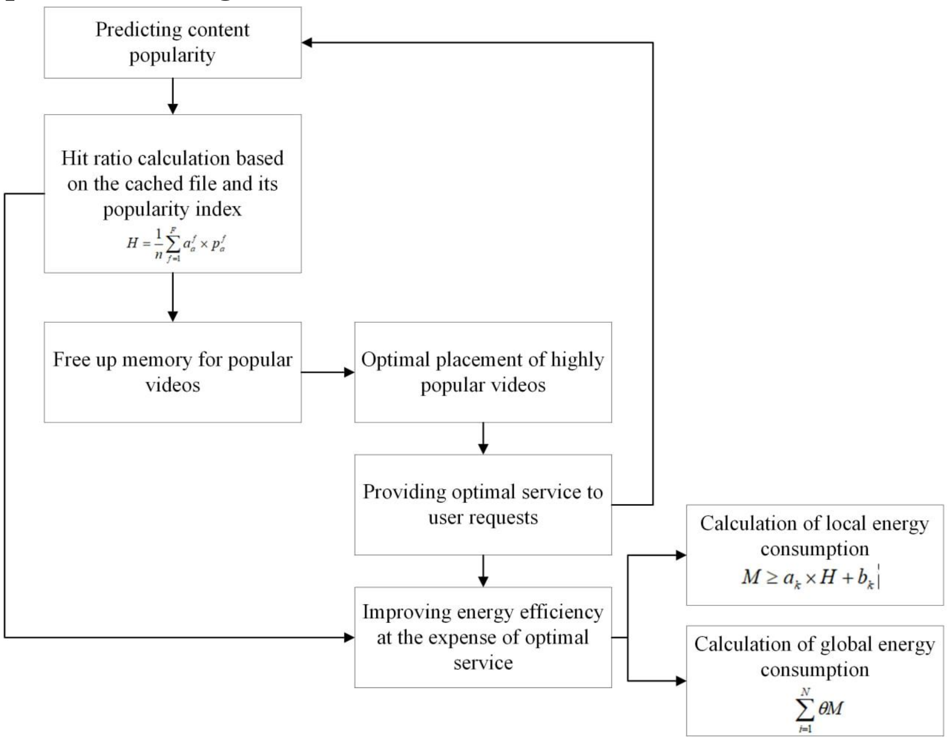
Predicting content popularity
Hit ratio calculation based on the cached file and its popularity index
$H=\frac{1}{n}\sum_{f=1}^{F}a_{a}^{f}\times p_{a}^{f}$
Free up memory for popular videos
Optimal placement of highly popular videos
Providing optimal service to user requests
Improving energy efficiency at the expense of optimal service
Calculation of local energy consumption
$M\geq a_{k}\times H+b_{k}$
Calculation of global energy consumption
$\sum_{i=1}^{N}\theta M$

Figure 3. Energy efficiency technique based on content popularity in CDN

As can be seen from the expression (7), all indicators except *M* are constant parameters and depend on the parameters of the data transmission network. For the most part, these parameters are fixed and almost impossible to change. The solution of the problem from this point of view is related to the problem of optimization of M_{it} - the amount of caches installed at time *t*. From this point of view, the main result of the mechanism presented in Figure 3 is to maximize the use of cache servers' memory capacity at the expense of optimal service to user requests and to increase the performance of cache servers to user requests. To achieve this, based on machine learning algorithms using user requests and attributes of video files, predicting the probability of video content becoming popular and storing videos with the highest popularity index on edge cache servers is caught [4-6].

A network consists of a central server, N:={1,...,N} cache servers, and a number of users associated with each cache. To simplify the process, we consider the case where each user is connected to a single cache server. Although only one central server is considered, the existing formula can be easily extended to a network infrastructure with multiple central servers. File requests from users are received on cache servers and served locally or forwarded to a central server depending on file availability. Each cache server stores only a portion of the entire content library. The purpose of this is to define the set of files that should be stored on the cache servers.

Below is a general CDN setup consisting of a central server and N cache servers. i – storage size of the cache server is denoted by M_i and is measured in bits. The central server has a library of files indexed as $f \epsilon F := \{1, \ldots F\}$. The size of an f file is defined in sf, which is also measured in bits. File sizes are collected into a $s\epsilon RF{+}{+}$ vector. The time is divided into slots and the time slot is indexed as t=1,2,.... The set of files stored in the i-cache at time t is denoted by $F_i(t)\epsilon F$. Alternatively, the storage vector $m_i(t)$ for the i-cache is defined as $m_i f(t) = 1$ for $f \epsilon F_i(t)$, otherwise zero. If the storage vector $m_i(t)$ satisfies $sTm_i(t) \leq M_i$, it is considered valid for i-cache. Let U_i be the set of users who send requests to i-cache. Since different servers may have limited network capacity and service ratio, it is assumed that users in U_i can receive C_c i bits and C_r i bits from the central server in each time slot from the i-cache at the maximum total rate in each time slot. The total cache request i for file f in each time interval t is given by $dif(t)$, which is measured in bits and summed into the vector $di(t)\epsilon RF+$. Given a given demand $dif(t)$, if $mif(t)$=1, the service policy performs service locally and redirects the request otherwise. Assuming that all requirements are subsequently satisfied, the total flow rate from the i-cache server is given by $mT_i(t)d_i(t)$, and the flow rate from the central server to users at U_i is given by the expression $1T\ d_i(t) - mT_i(t)d_i(t)$ is given, measured in bits in both timeslots.

If user requirements are known in advance, optimal content placement involves solving an integer programming problem in $m_i(t)$. The idea is to associate relevant costs with network flows and find the file placement that minimizes the overall cost. In the present context, we define the cost functions corresponding to the flows in the last mile connections connected to the i-cache server as χ_i:$R \rightarrow R$ and corresponding to the flows between the user connected to the i-cache server and the central server φ_i:$R \rightarrow R$ must be determined (blue dotted lines in Figure 3). On the other hand, for the topology presented in Figure 3, φ_i(.) represents the cumulative cost function that penalizes the flows in the backlinks and last-mile connections connected to the i-cache. Since the cost functions can usually be arbitrary, this approach is significantly more general and provides more flexibility than traditional displacement-based policies. As an example, the quadratic penalty $\chi_i(x) = c_1 x_2 + c_2$ imposes a base cost in addition to the square of the network usage, which prevents very high usage on any single server, thereby preventing buffer overflows or requests prevents the possibility of deletion. Another example is a simple linear cost that is multiplied by $\chi_i(x) = wx + 1$ 2x2, with a strict convex regularization in addition to penalizing large file downloads. Finally, the Kleinrock average delay function for a narrow link with capacity C_i is given by the expression $\chi_i(x) = x/(C_i - x)$ for $0<x<C_i$. In contrast, eviction-based policy parameters typically cannot directly control delays [7-8].

Flexibility in response to temporary changes in demand is also important in practice. For example, the requirements $\{d_{if}(t)\} \geq 1$ are usually independent and not uniformly distributed, but exhibit temporal decay and correlation. In particular, the popularity of content often decreases over time. For example, it has been observed that demands often have a diurnal approach and reach peak values at certain times of the day. A complete implementation of the content caching algorithm is shown in Figure 4.

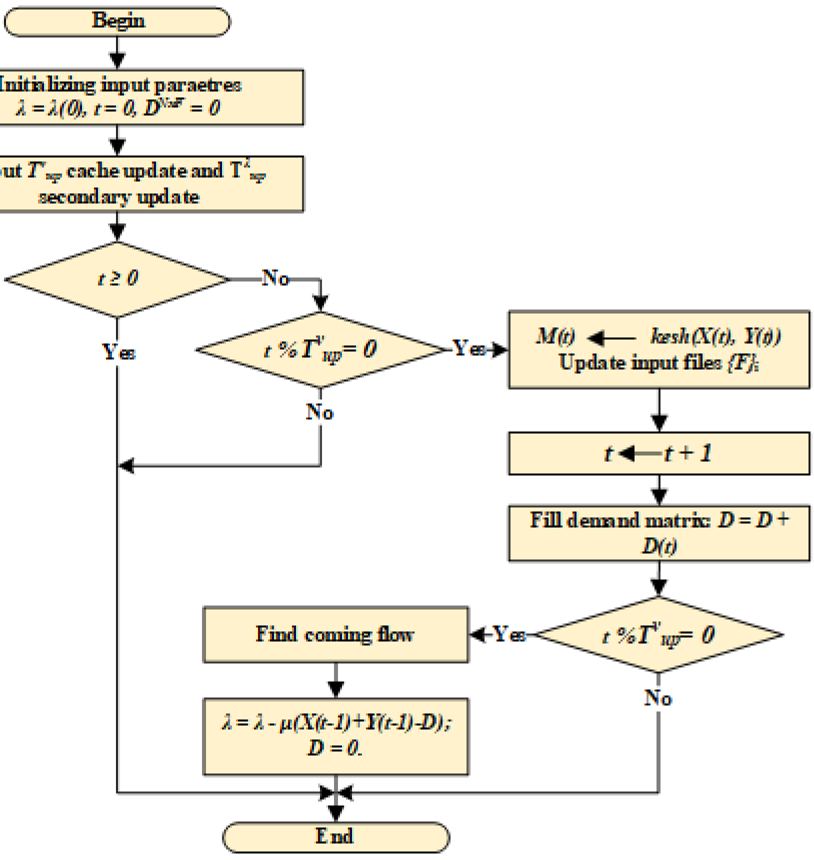
Begin
Initializing input paraetres
λ = λ(0), t = 0, D^{N×F} = 0
Input T'up cache update and T^λup secondary update
t ≥ 0
No
Yes
t % T'up = 0
Yes
No
M(t) ← kesh(X(t), Y(t))
Update input files {F}i
t ← t + 1
Fill demand matrix: D = D + D(t)
t % T^λup = 0
Yes
No
Find coming flow
λ = λ - μ(X(t-1)+Y(t-1)-D);
D = 0.
End

Figure 4. Content caching algorithm

In this case, the requirements are accumulated over time T^{λ} and used to fill the matrix $D(t)$. At each update interval, the expected stream variables $\{x'_{if}, y'_{if}\}$ are obtained using the current value of λ_{if} in the given algorithm, which is then updated. Expected streams are also used to update storage vectors $\{m_i(t)\}$. The function cache$(X(t), Y(t))$ indicates whether one of the placement policies is used. The cache refresh interval is determined by a separate size.

To get an additional boost on the performance of the algorithm, it is convenient to think of λ as the hidden cost of not satisfying the demand. Prices increase when demand is not satisfied by expected flows at any time t or when $x_{if}(t) + y_{if}(t) - d_{if}(t)$ is negative. Expected flows at key upgrade stages are adjusted based on higher or lower shadow prices. For example, if the demands are constant, the expected flows follow them so that the difference $x_{if}(t) + y_{if}(t) - d_{if}(t)$ is not very large.

As a simple example, consider a file *f* where the demand *i* from the user is zero for all time *t*. For such a file, if $l_{if}(0) = 0$, it remains zero for all time t. In fact, even though the initialization is different, the dual variables approach zero because $x'_{if}(t)$ and $y'_{if}(t)$ are fixed whenever $l_{if}(0) > 0$ will be positive. It can also be said that if a given $d_{if}(t)$ remains high for all t. Extending the argument, the double variable $l_{if}(t)$ "observes" the requirements of $d_{if}(t)$, although its observability is limited by the step size μ. Thus, the expected streams $\{x'_{if}(t), y'_{if}(t)\}$, which are monotone functions of $l_{if}(t)$, also track the popularity of files. In addition, occasional increases or decreases in demand have a minimal effect on the evolution of $l_{if}(t)$ due to the averaging effect inherent in the dual algorithm [8-9].

The interpretation presented here forms the basis of the proposed content placement algorithm. Expected streams represent the average popularity of files that can be used to host content online.

Results and Discussion

When serving users on a traditional CDN, after receiving a user request, the cache server checks whether the file exists as a local element. In case of a cache miss, the file is downloaded from the central server, served to the user, and stored on the cache server. When the cache is full, different migration algorithms are used, which are explained in detail.

In this work, in contrast to schemes based on permutation, a method using expected flows – $\{x_{if}(t)\}$ is used to obtain the memory allocation $m_i(t)$. Because pending streams are designed to meet average, not instantaneous, requirements, the resulting memory allocations typically do not contain every file that is eligible for a cache miss depending on the value of $x_{if}(t)$. This approach differs greatly from existing replacement algorithms such as LRU, where the out-of-cache file is always saved and the saved file is deleted.

Deployment of content within a shared first topology should be done periodically because of the high cost of retrieving the associated content. The refresh interval marked as tv up varies from hours to days, and files in the cache are updated at the beginning of each such interval. For this, another content placement algorithm has been developed, its implementation is shown in Figure 5.

American Journal of Interdisciplinary Research and Development
ISSN Online: 2771-8948
Website: www.ajird.journalspark.org
Volume 20, Sep., 2023

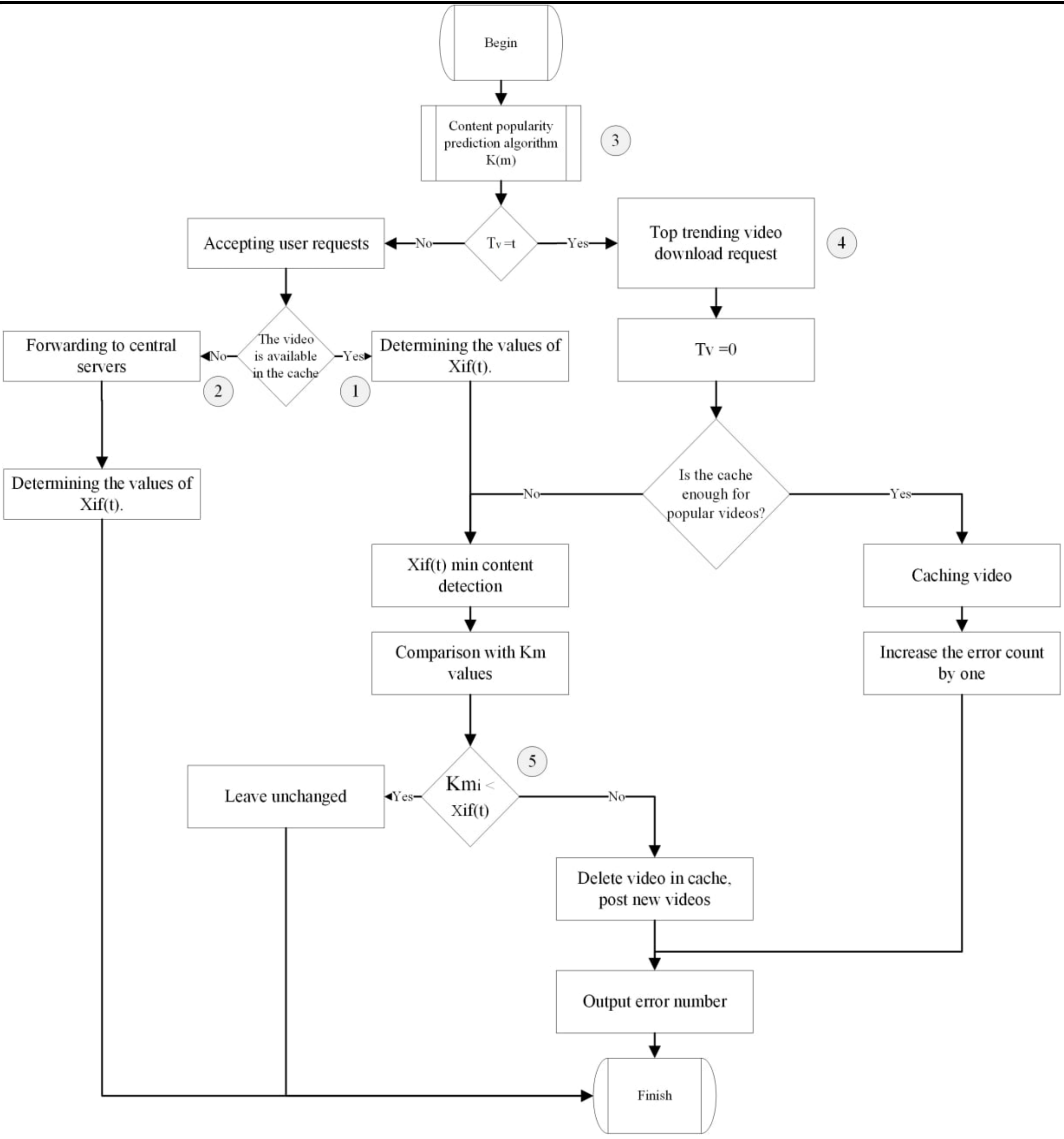
Begin
Content popularity prediction algorithm K(m)
3
Tv =t
No
Yes
Accepting user requests
Top trending video download request
4
The video is available in the cache
No
Yes
Forwarding to central servers
2
1
Determining the values of Xif(t).
Tv =0
Determining the values of Xif(t).
Is the cache enough for popular videos?
No
Yes
Xif(t) min content detection
Caching video
Comparison with Km values
Increase the error count by one
5
Leave unchanged
Yes
Kmi < Xif(t)
No
Delete video in cache, post new videos
Output error number
Finish

Figure 5. Content optimal caching algorithm based on artificial intelligence

Developed algorithm: First, files with high $x_{if}(t)$ values are considered popular and cached. That is, with each update, the cache ensures that the files with the largest $x_{if}(t)$ values match. In this case, the popularity of files is predicted based on artificial intelligence methods. As file sizes may vary, the number of saved files may also vary. If the list of files requested in a recent time interval is very large, only the files matching the largest values of $x_{if}(t)$ are placed in the cache edges.

In the developed algorithm, the cache server provides video content based on the requests received by the user. In this case, if the requested content is available in the cache (1) the video is presented from the cache, otherwise, the video is presented in the form of streaming from the central server through the cache (2). In both cases, the expected values of $x_{if}(t)$ of the streams of files in the cache are determined. At the same time, based on video attributes on the cache server and internal file attributes, content popularity prediction algorithms work on the central servers (3) and calculate the probable popularity coefficients Km of the existing files [9-10].
The cache servers operate without updating the stock of files for a pre-entered Tv up time, and at the specified time it sends a request to download the videos with the highest probability of being popular (4). Based on the size of the cache, the files whose popularity is higher than the value $x_{if}(t)$ of the files in the cache are downloaded, and the cache files are replaced based on the condition Km>$x_{if}(t)$ (5). This algorithm ensures that video content that can be requested by users many times is placed on the cache server and that future video requests are served based on the maximally limited caches. As a result, the volume of information exchange between users and central servers is minimized. This, in turn, allows you to increase the Hit ratio and, in turn, make maximum use of the available cache memory. Through this, the number of requests to the central server or external servers is reduced and the overall network energy saving is increased.

Energy consumption of CDN network elements

The main elements of a CDN network are a central server, routers, switches, transponders, amplifiers, multiplexers/demultiplexers, and edge cache servers. Therefore, in order to improve the energy efficiency of the CDN network, primary information about their energy consumption is needed. Below are the realistic energy consumption values of these elements.
Central server. Usually, server equipment manufacturers provide information about energy consumption in technical documents or specifications. Power consumption can be set for different operating modes, including maximum load (peak demand), normal load, or standby load.
Routers and their ports. The energy consumption of routers depends on several factors, such as: model/manufacturer, configuration, intensity of use, data transmission technologies. In general, the power consumption of a router ranges from a few watts (W) to several hundred watts, based on the above facts. As an additional note, the total power consumption of a router also depends on the number of ports required per wavelength. The number of ports varies depending on the wavelength division technology (TDM, WDM) used. Thus, the actual number of router ports required for each band wavelength depends on the router's capabilities. Watt is expressed as (8):
1W=B*A (8)

Based on the above information, the energy consumption of routers is determined from 5W to 12W per hour.

Switches and their ports. The energy consumption of switches is almost the same as the energy consumption of the router mentioned above. Therefore, their energy consumption is determined from 5W to 12W per hour.

Transponders. The energy consumption of transponders in the CDN network can vary significantly depending on the technologies used, models and the amount of data transmitted. On average, the power consumption of a transponder usually varies from several tens to several hundred watts. However, it should be noted that more accurate figures can only be provided by the manufacturer or seller, since the exact power consumption depends on the specific transponder model and configuration, as well as the current workload and operating conditions. For more accurate information on power consumption, it is recommended to refer to the specifications or technical documentation of a particular transponder. The energy consumption of common optical transponders does not exceed 200 watts per hour.

Amplifiers. The power consumption of amplifiers in a CDN network can also vary significantly depending on the specific conditions and characteristics of the device. Usually, the power consumption of the amplifier is from several tens to several hundred watts. However, exact values may vary greatly depending on factors such as amplifier type and model, amplification technologies used, device configuration and operating mode. For example, optical amplifiers used in fiber optic communication systems can have a power consumption of 10 to 100 watts per hour.

Multiplexer/demultiplexers. Power consumption of multiplexers and demultiplexers is measured in watts. This depends on various factors, including the specific model, technology and operating frequency of the device, as well as the number of I/O and data being converted. In general, the power consumption of a multiplexer/demultiplexer is 15 to 40 watts per hour.

Edge cache servers. The energy consumption of a marginal cache server is 100 to 250 watts per hour for a medium-sized company. Such a server can have two or four processors, the amount of RAM varies from 16 to 32 GB, and several hard drives from 1 to 4 TB each. A linear programming model can be implemented using the energy consumption data of the CDN network elements presented above.

The input values for the power consumption parameters used in the model are shown in Table 1. The power consumption of an 8-slot CRS-1 is 8000 W, including backplane power consumption, and therefore the power consumption of a 40 Gb/s port is estimated to be 1000 W. The power consumption of the Cisco ONS 15454 optical filter card is 16 W and the Cisco ONS 15501 optical amplifier is 8 W.

Table 1. Input values for power consumption parameters

The distance between two adjacent EDFAs (*S*)	80 (km)
The number of wavelengths in a fiber (*W*)	16
Wavelength capacity (*B*)	40 (Gbps)
Router port power consumption (P_p)	1000 (W)
Transponder power consumption (P_t)	73 (W)
EDFA power consumption (P_a)	8 (W)
Optical switch power consumption (P_o)	85 (W)
MUX/DEMUX power consumption (P_{md})	16 (W)
Cache server power consumption factor (∅)	7,4 (WpGB)
Probability of hitting the cache server	Avoid/fall/proposed method

The energy consumption of a 1-bit stream (READ/WRITE operation) is given as 211×10^{-9} J. This value is converted to the power (watts) used to transfer 1 GB of data over a certain period of time for use in the analytical model. Energy consumption for streaming 1 GB of data: $211\times10^{-9}\times10^{9}\times8=1688$ J. A typical maximum access speed for a Fiber Channel hard drive is up to 100 MBps. However, typical average data rates are about 50% of this value and reach up to 50 MBps. Increasing the access speed of the hard disk will reduce the cache refresh time, but will result in more power consumption. To achieve average power consumption, a data transfer rate of less than 35 Mb/s is considered on the hard disk. Time required to stream 1 GB: $10^3\times8/35=228$ seconds. So, the power consumption for 1 GB of caching is 1688/228 = 7.4 WpGB.

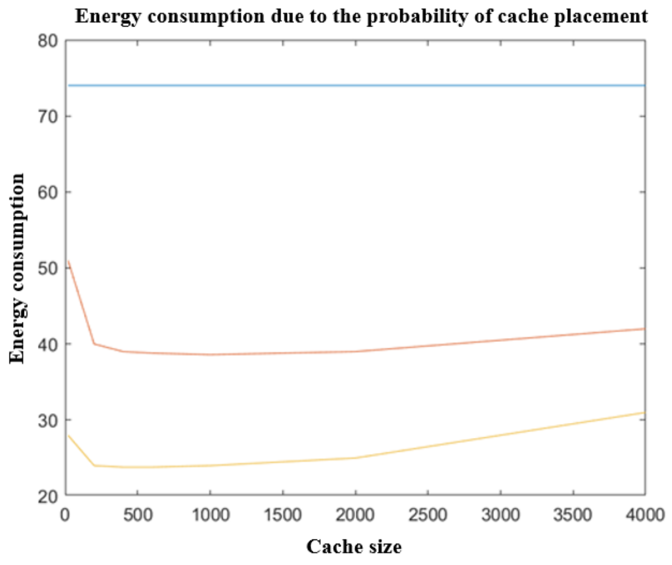

Figure 6. Energy consumption as a percentage of whether user requests are cached or not

The result in Figure 6 is realized by the analytical model, which takes into account the variation of the cache index. The change in the cache indicator shows the probability that users will not access the content requested by the local cache server, will access it (existing methods), and will access it through the proposed method. The result is the energy consumption due to increasing the size of the local cache server, determined by whether or not the cache is dropped. According to the obtained results, a steady 74 watts of power consumption in non-dropping, 38-51 watts of power consumption with existing methods with a cache server, and 28-31 watts of power consumption with the proposed method were achieved. The efficiency of the proposed method is 40% better than the existing methods.

Conclusion

A mechanism for efficiently placing content in CDN caches and an algorithm based on artificial intelligence have been developed. This mechanism and an algorithm based on artificial intelligence aim to improve the efficiency of content placement in CDN caches, taking into account several factors such as content popularity, user location and server availability. The mechanism consists of the following steps: content popularity analysis; analyze user location; analysis of server availability; and applying an algorithm based on artificial intelligence. In content delivery networks, the number of requests to central or external servers and the total energy consumption of the network have been reduced through the mechanism and algorithm of effective caching of video content based on artificial intelligence. Analysis of the energy consumption of the elements of the CDN network was carried out, based on the results of the analysis, calculations of the energy consumption were carried out, taking into account whether the user's requests were delivered to the cache server or not. The result is the energy consumption due to increasing the size of the local cache server, determined by whether or not the cache is dropped. According to the obtained results, a steady 74 watts of power consumption in non-dropping, 38-51 watts of power consumption with existing methods with a cache server, and 28-31 watts of power consumption with the proposed method were achieved. The efficiency of the proposed method is found to be 40% higher than the existing methods.

References

1. V. Mathew, R. K. Sitaraman, and P. Shenoy, “Energy-aware load balancing in content delivery networks,” in 2012 Proceedings IEEE INFOCOM, 2012, pp. 954–962.
2. W. Wang et al., “CRCache: Exploiting the correlation between content popularity and network topology information for ICN caching,” in Communications (ICC), 2014 IEEE International Conference on, 2014, pp. 3191–3196.
3. Liangzhong Yin and Guohong Cao, “Supporting cooperative caching in ad hoc networks,” IEEE Trans. Mob. Comput., vol. 5, no. 1, pp. 77–89, 2006.

4. G. P. Lorenzo Saino, Ioannis Psaras, “Icarus: a Caching Simulator for Information Centric Networking (ICN),” in . ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)., 2014, pp. 66–75.
5. A. Compagno, M. Conti, C. Ghali, and G. Tsudik, “To NACK or not to NACK? negative acknowledgments in Information-Centric Networking,” in Computer Communication and Networks (ICCCN), 2015 24th International Conference on, 2015, pp. 1–10.
6. W. K. Chai, D. He, I. Psaras, and G. Pavlou, “Cache 'Less for More' in Information-Centric Networks,” pp. 27–40, 2012.
7. Xiaohu Chen, Qilin Fan, and Hao Yin, “Caching in Information-Centric Networking: From a content delivery path perspective,” in Innovations in Information Technology (IIT), 2013 9th International Conference on, 2013, pp. 48–53.
8. Y. Wu, Y. Jiang, M. Bennis, F. Zheng, X. Gao, and X. You, “Content popularity prediction in fog radio access networks: A federated learning based approach,” in 2020 IEEE International Conference on Communications (ICC), Jun. 2020, pp. 1–6.
9. Abdorasoul Ghasemi, Amirhosein Ahmadi, Cache management in content delivery networks using the metadata of online social networks, Computer Communications, Volume 189, 2022, Pages 11-17, ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2022.02.021.
10. Y. Tao, Y. Jiang, F. Zheng, M. Bennis, and X. You, “Content popularity prediction in fog-rans: A bayesian learning approach,” in 2021 IEEE Global Communications Conference (GLOBECOM), Dec. 2021, pp. 1– 6.