A New Cache Placement Strategy for Wireless Internet of Things

Hua Wei, Hong Luo, Yan Sun

Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia,
Beijing University of Posts and Telecommunications, China
weihua2015@bupt.edu.cn, luoh@bupt.edu.cn, sunyan@bupt.edu.cn

Abstract

Caching has shown the success in performance improvement for many wireless communications and networking systems. However, the existing researches generally decide whether cache the data or replace it rely on local content popularity on each single node. It will cause different nodes caching the same data and result in unnecessary cache redundancy. In this paper, we investigate the global optimal problem of cache placement for IoT. We first prove that finding the optimal data cache location from the whole network is an NP-hard problem, and propose a centralized algorithm to obtain the approximate global optimal solution based on the Lagrangian Heuristic Algorithm. Then, inspired by the Lagrangian relaxation, we transform the iteration procedure of finding the optimal cache location into local decisions of cache location selection and cache replacement, and we propose a distributed cache placement algorithm. Besides, the cache replacement algorithm can also be used to adjust the best cache location when the user requirement changes. Finally, we implement the distributed cache placement strategy in NDN. The experimental results show that the distributed caching strategy approximates the global optimal solution very well, and can save the network traffic by about 12.6% on average comparing with other caching strategies.

Keywords: Information Center Network (ICN), Internet of Things (IoT), Lagrangian Heuristic Algorithm (LHA), Cache placement, Cache replacement

1 Introduction

In the traditional wireless communications and networking systems, it takes the TCP/IP architecture as the core, and the packets are forwarded according to the IP address in the packet header regardless of the data content. Different with it, in the Information-centric network [1-2], the data not rely on IP address but data name to be forwarded. The data name is the only identifier of the information. And it has a caching mechanism which can cache data at the routing node, so that the data will be independent of the physical location. The ICN architecture can significantly reduce the data transmission, improve security and network mobility [3]. These features are useful for many IOT applications, for example, production process monitoring, logistics data transmission and so on. A typical Information-Center Network is shown in Figure 1. The user requests the data by sending an *Interest* packet. Obviously, the transmission cost and delay are small, when the cache node is close to the user. However, it will cause the same data has been cached many times, especially when different access points request the same data.

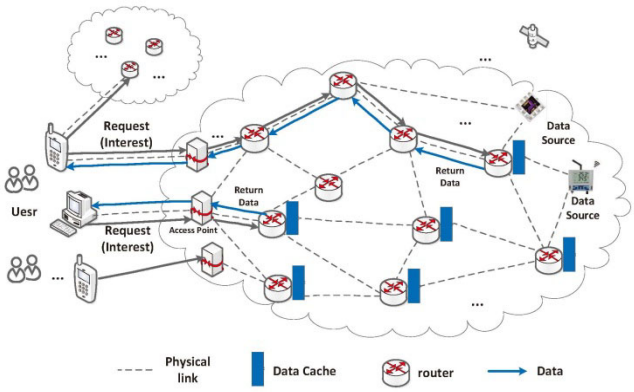


Figure 1. A typical Information-Center Network

The ICN architecture which is suitable for traditional Internet is also suitable for IoT. However, the IoT still has some different characteristics with the traditional Internet. For example, in the IoT, the node resources and transmission bandwidth are limited. Therefore, the caching algorithm in the IoT must adapt to its characteristics when using the ICN architecture. In the most of existing research, the data in the cache node is independent, that is, each node is based on its local content popularity to determine whether cache the data or not. Therefore, the data which has high content popularity might be cached in multiple nodes, it means the cache redundancy is high. Due to the limitations of cache size in the IoT, it is necessary to design a better cache strategy from the view point of global optimization for users to avoid the highly cache

*Corresponding Author: Hua Wei; E-mail: weihua2015@bupt.edu.cn
DOI: 10.3966/160792642019052003006

redundancy. Meanwhile, to improve the network performance and reduce the user's waiting time when it accesses the data, we also need to set the cache point to the end user as close as possible so as to provide the near-end service. So the challenge of caching strategy in IoT is how to balance the service quality and the network limitation and find the global optimal solution.

In order to solution this problem, researchers have proposed many methods, such as virtual SDN algorithm and hierarchical optimization algorithm. However, these methods aren't suitable for IoT environments and have low performance. Different with previous works, we study the problem from two aspects: We first model the network structure and define the problem as minimizing the total network transmission consumption with minimum cached data. Then, the problem is converted to Generalized Facility Location Problem (GFLP). In order to calculate the cache position in the network, a Lagrangian heuristic algorithm is used to find the global approximate optimal solution [4]. Secondly, we propose a distributed cache placement and replacement algorithm based on the idea of Lagrangian relaxation which has been used in finding the global approximate optimal solution. The distributed algorithm is executed on each routing node with local information while ensuring that there is no excessive cache redundancy. Hence it is effective when the global information can't be counted in real time. After that, we take the well-known Named Data Network as an example, and figure out the specific implementation process of the above algorithm [5]. Finally, we use NDNSIM platform to make simulation experiments [6-7], the results show that the average similarity between the distributed strategy and the global approximate optimal solution is 88.7%. Therefore, we conclude that the result of the distributed caching strategy can simulate the global approximate optimal solution. The main contributions of this paper lie in the following three aspects.

- We define the data cache problem as minimizing the network transmission consumption and convert to generalized facility location problem. Since the problem belongs to the NP-hard problem, we use the Lagrangian heuristic algorithm to solve it and obtain the approximate global optimal solution.
- We propose a distributed cache placement strategy which only uses local information to make the cache decision while ensuring that there is no excessive cache redundancy.
- We provide the implementation of the proposed DCA in the Named Data Network and compare the results of the distributed caching strategy and the global approximate optimal solution through a large number of experiments. The results show that the distributed algorithm is 88.7% approximating to the global algorithm and outperforms other distributed algorithms.

The rest of the paper is organized as follows. The section II is related work. We give the problem formulation in section III. In Section IV, we describe optimal cache position calculation in detail, and Section V describes the distributed caching strategy. In Section VI, we describe how to implement the specific process of achieve the distributed caching strategy in NDN. The experiment and performance evaluation is show in Section VII. Finally, conclusions and future work are given in Section VIII.

2 Related Work

In order to solve the problem of repeated data transmission in existing networks, the ICN's pioneer RTIAD proposed Information-Centric Network (ICN) architecture [1]. Many researches which based on the ICN architecture have achieved academic support and recognition. They have different architectural models and focuses [8], in which the study of Named Data Network project has received wide attention [5].

ICN in-network caching strategies are based on two caching models: On-path and Off-path caching [9]. On-path caching is implemented by storing data on the forwarding path, and most of caching algorithms belong to on-path caching. When the node doesn't have enough storage space, we need a suitable cache replacement algorithm to replace the inefficient content with the efficient one. At present, the existing algorithms include Least Recently Used (LRU), First in First out (FIFO), virtual SDN algorithm [10], hierarchical optimization algorithm [11], Content popularity prediction algorithm [12], MAGIC algorithm [13] and so on.

The LRU and FIFO are the commonly used replacement algorithms, it uses short-term local history records to sort the content, and determines the cache priority of the content by the sort. As the short-term history does not completely reflect the regularity of future access [14], it may cause invalid replacement of content. Sun et al. considered that it incurs faster cache replacement and degrades the cache performance, since the same data is replicated in all routers along the request path [11]. Therefore, they proposed a hybrid cache strategy to overcome the drawback.

Charpinel et al. [10] proposed a SDN method that can provide a programmable forwarding strategy and a caching strategy for CCN. This method eliminates the need for mapping between content names and identifiers. And it added a Cache Rules Table (CRT) to storing cache rules which inform what kind of content should be stored in cache. They achieved the above method through SDCCN. In SDCCN, the data plane doesn't need to recognize any frame format and the processing flows is similar to OpenFlow. Liu et al. proposed a Content Popularity Prediction method to achieve the popularity prediction, which is based on computing resources and links in the SDN [15]. However, since the data layer and the control layer

need to communicate through the Protocol-Oblivious Forwarding (POF) protocol, the excessive communication cost is increased. And the SDN approach increases the control layer, it is not suitable for the IOT environment.

Ren et al. [13] proposed a distributed MAGIC (MAx-Gain In-network Caching) algorithm that use less inter-node communication to reduce cache redundancy, and it is designed for the IoT environment. Hence, it has a significant improvement in wireless networks with limited cache size at each wireless node. However, the method still has many problems. For example, the minimum penalty value may have changed before the data back. Then, it obtained conclusions by experiment just with several nodes. There is no theoretical proof of the effectiveness of the algorithm. Finally, it's important to note that MAGIC algorithm will cause some nodes storing a series of hot data which may reduce the network performance.

Content popularity algorithm predicts the popularity ranking by calculating the local popularity. It replaces the content by ranking order. Wu et al. proposed a probability based heuristic caching strategy, when the data packet is returned back, on-path cache nodes decide whether to cache the content with certain probability [16]. Zhang et al. presented a Popularity Prediction Caching replacement method for chunk-level cache by discovering the relevance among video chunks in ICN from the perspective of user watching behavior [17]. All of these methods have the following two problems. Firstly, the local prediction results are not accurate enough. Secondly, it may cause many nodes in the network storing the same content.

3 Problem Formulation

In this section, we first formally define the network topology and transmission consumption. Then, we formulate the problem as minimizing the total transmission consumption with minimum cached data.

It should be noted that we use the terms "routing node", "caching node", "router" and "node" interchangeably to refer to cache-enabled network devices [18]. Furthermore, we refer to content "data", "contents" and "data packet" interchangeably to refer to the cacheable unit.

3.1 System Model

Let *G(V, E)*, a fully connected undirected weighted graph, where *V* denotes the set of nodes in the network and *E* denotes the set of communication links between nodes. The weight W{e|e ∈ E} indicates the bandwidth of the link *e*. Nodes are divided into three types: *Producer (P)*, *Router (R)* and *Access Point (A)*. P = {1, 2, ... m, ..., M} is the set of producers representing the nodes which are data sources. R = {1, 2, ... i, ..., N} is the set of routers representing the nodes which can forward and cache data. A = {1, 2, ... j, ..., Q} is the set of access points representing the nodes which gather the user's query and request the data. For simplicity, let *P (m)* be the m-th data source node which the number is m. Similarly, let *R(i)* and *A(j)* be the i-th router node and j-th access point, respectively. Obviously, $V = P \cup R \cup A$.

Suppose that there are k-types of data in the network. The data types are indexed by k ∈ K and the data source node m produce the k-type data. x_{ijk} indicates whether the type-k data requested by the j-th AP node is cached in the router node *i*.

$$x_{ijk} = \begin{cases} 0, \textit{Not being cached} \\ 1, \textit{Being cached} \end{cases} \quad \textbf{(1)}$$

In order to minimize the cached data in the network, the same data type on a single link can only be cached once, *i.e.*

$$\sum_{i \in R} x_{ijk} = 1,\ j \in A, k \in K \quad \textbf{(2)}$$

Assume that the size of each data type is same and denoted by *s*. Let *s* be the size of the data. The U_i be the size of cache space at the router node *i*.

$$\sum_{k \in K} x_{ijk} * s \le U_i,\ i \in R,\ j \in A \quad \textbf{(3)}$$

The communication cost between node *i* and node *j* is denoted by c_{ij}. And c_{ijk} represents the communication cost of transmitting type-*k* data between node *i* and node *j*.

$$c_{ijk} = d_{ij} * f_{ijk}, \quad i \in R,\ j \in A, k \in K \quad \textbf{(4)}$$

where d_{ij} is the distance between node *i* and node *j*, f_{ijk} is the request frequency of node *j* for type-*k* data at node *i*.

And we use the $c_{mi}(k)$ define the fixed cost which is transfer the type-*k* data from data source node m to the router node *i*. Now, we have the second constraint of link capacity.

$$\sum_{k \in K} \sum_{j \in A} f_{ijk} x_{ijk} \le \frac{W_e}{s},\ i \in R, e \in E, e = \{i, j\} \quad \textbf{(5)}$$

We use a Boolean factor y_i to indicate whether the cache space is full or not, and set $y_i = 1$ if it is full; otherwise, $y_i = 0$.

$$\sum_{i \in R} y_i \le N, i \in R \quad \textbf{(6)}$$

The cache cost *P* is the added transmission consumption due to cache replacement.

$$P = Min\sum_{j=A} f_{ijk} * d_{mi}, i \in R, k \in K \quad (7)$$

Here, type-*k* data is produced by data source node *m*, and d_{mi} represents the distance from data source node *m* to router node *i*. For simplicity, we assume that the routing interface will not be changed and data request frequency is stable. Meanwhile there is no packet loss in the network.

3.2 Problem Definition

Based on the discussed above, we can calculate the total costs in the network. The aim is to minimize the cost with the constraints (1) to (7). The objective function can be formulated as Equation (8).

$$Min\sum_{i\in R} Py_i + \sum_{i\in R}\sum_{k\in K} c_{mi}(k) + \sum_{i\in R}\sum_{k=K}\sum_{j\in A} c_{ijk}x_{ijk} \quad (8)$$

subject to $$\sum_{i\in R} x_{ijk} = 1,\ j \in A, k \in K \quad (9)$$

$$\sum_{k\in K} x_{ijk} * s \le U_i, i \in R,\ j \in A \quad (10)$$

$$c_{ijk} = d_{ijk} * f_{ijk}, i \in R,\ j \in A, k \in K \quad (11)$$

$$\sum_{k\in K}\sum_{j=A} f_{ijk}x_{ijk} \le \frac{W_e}{s}, i \in R, e \in E, e = \{i, j\} \quad (12)$$

$$P = Min\sum_{j\in A} f_{ijk} * d_{mi}, i \in R, k \in K \quad (13)$$

$$c_{mi(k)} = 1 * d_{mi}, i \in R, k \in K \quad (14)$$

$$x_{ijk} \in \{0,1\}, i \in R,\ j \in A, k \in K \quad (15)$$

$$\sum_{i\in R} y_i \le 1,\ y_i \in \{0,1\}, i \in R \quad (16)$$

where the *P* is the replacement cache cost, d_{mi} is the fixed data transmission costs and c_{ijk} is variable transmission costs.

4 Optimal Cache Position Calculation

Facility location problem is to find locations for new facilities such that the conveying cost from facilities to customers is minimized. Consider taking cache cost *P(k)* as fixed site setup cost, fixed data transmission cost c_{mi} as facility setup cost and variable transmission costs c_{ijk} as the connection cost of satisfying the type-*k* demand of customer *j* by the facilities in site *i*. Then, the problem is converted to Generalized Facility Location Problem (GFLP). As we know, the classical NP-hard problem Capacitated Facility Location Problem (CFLP) is a special case of GFLP with $c_{mi} = 0$. GFLP is an NP-hard problem obviously.

Wu et al. proposed an efficient Lagrangian heuristic algorithm (LHA) to solve the GFLP [4]. In this paper, firstly, we use the Lagrangian relaxation method to estimation the lower bound and the upper bound [19-20]. Then, we implemented the classical subgradient method to find the optimal Lagrangian multipliers [21-22]. Finally, we list the complete Lagrangian Heuristic Algorithm (LHA) to calculate the cache location.

4.1 The Lower Bound and the Upper Bound

We relax constraints (12) in objective function (8) with multipliers λ_i, $i \in R$. Then we get a Lagrangian relaxation of the objective function (8) as Equation (17).

$$Min\sum_{i\in R} Py_i + \sum_{i\in R}\sum_{k\in K} c_{mi}(k) + \sum_{i\in R}\sum_{k\in K}\sum_{j\in A} c_{ijk}x_{ijk} + \sum_{i\in R}\lambda_i\left(\sum_{k\in K}\sum_{j\in A} f_{ijk}x_{ijk} - \frac{W_e}{s}\right) \quad (17)$$

subject to (9)-(11) and (13)-(16).

Then, we can decompose the above problem into *N* subproblems, one for each routing node, by leaving constraints (16) aside.

$$Min\ Py_i \sum_{k\in K} c_{mi}(k) + \sum_{k\in K}\sum_{j\in A}(c_{mi} + \lambda_i f_{ijk})\, x_{ijk} \quad (18)$$

subject to (9)-(11) and (13)-(15). For each subproblem, y_i is either equal to 0 or 1. If $y_i = 0$, then $Py_i = 0$. Then the objective function (18) can be further decomposed into *K* subproblems, one for each data type. The objective function is

$$Min\ c_{mi}(k) + \sum_{j\in A}(c_{ijk} + \lambda_i f_{ijk})x_{ijk} \quad (19)$$

Subject to $\sum_{k\in K} x_{ijk} * s \le U_i, \quad i \in R,\ j \in A$

$$c_{ijk} = d_{ij} * f_{ijk}, \quad i \in R,\ j \in A, k \in K$$

$$c_{mi}(k) = 1 * d_{mi}, \quad i \in R, k \in K$$

$$x_{ijk} \in \{0,1\} \quad i \in R,\ j \in A, k \in K$$

For each data request frequency f_{ijk}, the problem (19) is an integer knapsack problem that can be solved by the greedy algorithm. Then, we can get the optimal solution $R^*(x_{ijk})$ and the minimal objective function value L^*.

Let L^*_{ik} denote the minimal objective function value of the subproblem (19), then the minimal objective function value of the subproblem (18) is

$$L_i^* = \begin{cases} P + \sum_{k \in K} L_{ik}^*, y_i = 1 \\ \sum_{k \in K} L_{ik}^*, \quad y_i = 0 \end{cases} \tag{20}$$

Finally, the minimal solution can be obtained by adding the constraints (16) after solving the N subproblem. The solution to the objective function (17) provides a lower bound to the original problem.

In order to get the upper bound of the problem, we solve the following linear programming problem to find a feasible solution of the original objective function (8).

$$Min \sum_{i \in R^*} \sum_{j=A} \sum_{k=K} c_{ijk} x_{ijk} \tag{21}$$

subject to $\sum_{i \in R} x_{ijk} = 1, \quad j \in A, k \in K$

$$c_{ijk} = d_{ij} * f_{ijk}, \quad i \in R^*, j \in A, k \in K$$

$$\sum_{k \in K} x_{ijk} * s \le U_i, \quad i \in R^*, j \in A$$

$$x_{ijk} \in \{0,1\} \quad i \in R^*, j \in A, k \in K$$

The optimal solution is an upper bound on the original problem. If it is smaller than the existing upper bound, the upper bound is improved.

The solution for objective function (17) not always be feasible to the original problem. In other words, the total cost defined by the set of router node R^* and x^*_{ijk} is smaller than the total demand. We can use the simple greedy heuristic to adjust the R^* and x^*_{ijk} ijk so that all demands can be satisfied. By this way, the upper bound U can be get.

4.2 Lagrangian Heuristic Algorithm

In order to update the Lagrangian multipliers, we use the classical subgradient method [21-22]. First, the subgradients and step size are calculated by Equation (22) and (23).

$$\psi(i) = \sum_{i \in R} x_{ijk} - 1, \; j \in A, k \in K \tag{22}$$

$$\mu = \frac{\sigma(U - L)}{\Sigma_{j \in A} \Sigma_{k \in K} \psi(i)^2} \tag{23}$$

where U is the upper bound and L is the optimal lower bound solved above. σ is a customize parameters with initial value between 0 and 2. The Lagrangian multipliers λ_i can be calculating by Equation (24).

$$\lambda_i = \lambda_i + \psi(i) * \mu \tag{24}$$

The complete Lagrangian heuristic algorithm is illustrated in Algorithm 1. Let *G(V,E)* denote the network topology, *MAXT* denote the maximum number of iterations and ε denote the allowed maximum interval between the lower and upper bounds. After iterative calculation, the set of optimal solution R^* and x^*_{ijk}, the lower bound L and the interval between the lower and upper bounds *GAP* can be obtained.

Algorithm 1. Lagrangian Heuristic Algorithm

Input: *G(V,E), MAXT,* ε

Output: R^*, x^*_{ijk}, *L, GAP*

```
1.  Set: LB= −∞ , UB=+ ∞ , GAP=+ ∞ ;
2.  Initialize λ_i^t = 0, t = 0 ;
3.  while t < MAXT and GAP < ε do
4.     Solve objective function (17) with parameter
       λ_i^t ;
5.     Get the optimal objection value L;
6.     Get the set of R* and x*_ijk ;
7.     if L > LB then
8.        LB = L;
9.     end if
10.    Solve linear programming problem (21);
11.    Get the optimal objection value U;
12.    if U < UB then
13.       UB = U;
14.    end if
15.    if (UB – LB)/LB ≤ GAP then
16.    GAP = (UB – LB)/LB;
17.    end if
18.    t = t + 1;
19.    Update Lagrangian multipliers λ_i^t ;
20. end while
```

5 Distributed Caching Strategy

The heuristic algorithm can find the approximate global optimal solution. However, we must obtain the global network topology and every node's request frequency when we use it. It is difficult to achieve in real networks. In order to achieve the approximate optimal solution in the IoT environment, inspired by the idea of Lagrangian relaxation [23-24] which has been proposed in the previous section, we propose a distributed cache placement algorithm (DCA).

The brief approach is as follows. Firstly, we use the Lagrangian relaxation process to decompose the distributed optimization problem into minimum transmission costs calculation in a single node. Then,

the minimum transmission costs problem is converted into maximizing the number of request satisfied by once transmission. Finally, select the best cache location on a transmission path. In the process of best cache location selection, if the selected cache is full, we execute a cache replacement procedure. This cache replacement procedure can also be used when new user requests come in.

5.1 Distributed Optimization Problem

The goal of a distributed optimization problem is to find at most one cache node on the path between the data request node and the data source node, which can minimize the network transmission costs.

Let $e^*(m, j)$ represent the path between the data source node $P(m)$ and the data request node $A(j)$.

$$e^*(m, j) = Path\{P(m), 1, 2, \ldots, i, \ldots, I, A(j)\} \quad \textbf{(25)}$$

where the node from 1 to I denotes all the router nodes on the path $e^*(m, j)$.

The objective function of distributed optimization problem is also can be formulated as Equation (8) with the constrain $i \in e^*(m, j)$ Similar to the idea of Equation (18), we decompose the problem into I subproblems. Then, similar to the idea of Equation (19), the distributed optimization problem can be further decomposed into K subproblems. The transmission costs about type-k data on the path $e^*(m, j)$ can be formulated as Equation (26).

$$Py_i + d_{mi} + \sum_{j \in A} d_{ij} f_{ijk}, i \in R \quad \textbf{(26)}$$

5.2 Problem Transformation

Let $g(k)$ denote the benefit value of cache type-k data. We define the benefit value as the number of request satisfied by once transmission.

$$g(k) = \frac{\Sigma_{j \in A} f_{ijk}}{\overline{d_{ij}}} \quad \textbf{(27)}$$

When the cache space is full, we set the Boolean factor $y_i = 1$; otherwise, $y_i = 0$. Let k' denote the data type which will be replacement when $y_i = 1$. In the next section, we give a detailed process of how to choose k'. Furthermore, the value P can be converted to minimizing the value g(k'). The value of g(k') y_i is calculated by Algorithm 3.

$$Py_i = \min \sum_{j \in A} f_{ijk} * d_{mi} \Rightarrow \min g(k') y_i \quad \textbf{(28)}$$

Then, the last two terms of equation (26) can be converted as follows:

$$\min(d_{mi} + \sum_{j \in A} d_{ij} f_{ijk}) \Rightarrow \max = \frac{\Sigma_{j \in A} f_{ijk}}{\overline{d_{ij}}} \quad \textbf{(29)}$$

where $\overline{d_{ij}}$ denotes the average distance from the cache node i to all the access point.

Finally, the benefit of cache type-k data in node i can be formulating as Equation (30).

$$\left[\frac{\Sigma_{j \in A} f_{ijk}}{\overline{d_{ij}}} - g(k') y_i \right] * e^{\lambda(1-\alpha_i)}, i \in R, k \in K \quad \textbf{(30)}$$

where λ is a parameter that determines the impact of bandwidth on the benefits, α_i is the bandwidth usage ratio of link $e(i, i + 1)$, Let *Tra* represent the number of data transmission which can be monitored during transmission. α_i can be calculated by the Equation (31):

$$\alpha_i = \frac{s * Tra}{W_{e(i,i+1)}} \quad \textbf{(31)}$$

5.3 Best Cache Location Selection

Our goal is to find the best cache location with the maximum benefit value. Aiming at comparison of the benefit value of each node on the path in a distributed way and lower the unnecessary control cost, the comparison is occurs when the interest message is forwarded.

The distributed cache algorithm is illustrated in Algorithm 2. Let $e^*(m, j)$ denote the network topology, *Interest* denote the data request packet and λ denote the parameter. After network transmission, the *data* back from data source m and the maximum value of benefit *MAX* can be obtained. If *MAX* > 0, the data also be cached in the best location; otherwise, the data would not be cached.

Algorithm 2. Distributed Cache Selection Algorithm

Input: $e^*(m, j)$, *Interest*, λ.
Output: *data*, *MAX*.

```
1.  Set: MAX=0;
2.  Send Interest from j to m.
3.  if Router node m receive an interest then
4.      if Cache space is full then
5.          Set y_i = 1 and Calculation g(k') by
        Algorithm 3;
6.      else
7.          Set y_i = 0;
8.      end if
9.      Calculation benefit value (BV) by
        Equation(30) with parameter λ
10.     Storage BV in the node;
11.     if BV > MAX then
12.         MAX=BV;
13.     end if
14.     Forward Interest to next node;
15. end if
16. if Router node m receive a data then
```

```
17.    if MAX == BV then
18.        Cache data;
19.    end if
20.    Forward data to next node;
21. end if
22. if Data source m receive an Interest then
23     Get the value MAX;
24.    Send data & MAX from m to j;
25. end if
```

Then, we propose a cache replacement strategy to calculate which type of data will be replaced. The strategy is based on the value of g(k).

The cache replacement algorithm is shown in Algorithm 3. Let C_i denote the set of cached data in node *i*. Calculate the function value $g(a)$ for all the elements in C_i. Let G_i denote the set of calculate result $g(a)$ in node *i*. Then, we have to sort all the elements in G_i. The minimum value is denoted as g(k'). Compare g(k') with the value g(k), if g(k') < g(k), replacement the data k' with data k.

Algorithm 3. Cache Replacement Algorithm

Input: $k,\ C_i$.

Output: k'.

```
1.    Set: Min=+∞
2.    Data k return to node i.
3.    Calculate the function value g(k);
4.      if Cache space is full then
5.          for all a ∈ C_i do
6.              Calculate the function value g(a);
7.              if g(a) < MIN then
8.                  MIN = g(a);
9.                  k'
10.             end if
11.         end for
12.       if g(k) >MIN then
13.           Remove Data k';
14.         Cache Data k;
15.     end if
16.   else
17.       Cache Data k.
18.   end if
```

6 Implementation in NDN

In this section, we take NDNSIM [5], the well-known Named Data Network project as the platform, discuss the specific process of cache location calculation and cache replacement strategy. In the NDN, communication is driven by the receiving end, i.e., the data consumer. In order to receive data from data producer, a consumer sends out an **Interest** packet. The router forwards the Interest packet relies on the data name instead of network address. A router remembers the interface from which the request comes in, and then forwards the Interest packet. Once the Interest reaches a node that has the requested data, a **Data** packet is sent back, which carries both the name and the content of the data. This Data packet traces in reverse the path created by the Interest packet back to the consumer [5].

Aiming at realizing the distributed cache algorithm, we modify the originally NDN model by adding the history access information to store access information. And we modify the Interest and the data packet format to transmit the node's benefit value between nodes.

Interest: The originally Interest message contains Content Name, Selector, Nonce. In order to transfer the maximum benefit value, a new field MAX is added in the Interest. When a router receives the Interest, it calculates the benefit value with Equation (30) and use the Algorithm 2 deal with Interest.

Data Packet MetaInfo: The data packet contains four parts, including Name, MetaInfo, Content and Signature. The original structure of MetaInfo in NDN includes Content Type, Freshness period, etc. Now, a new field MAX is added in the Data Packets MetaInfo, too. When a data producer receives the Interest, it gets the *MAX* value from it. Then, it set the field MAX with the same value meanwhile send back the data packet. The router compares the MAX with local value when it forwards a data packet.

Access Information Table (AIT): The Access Information Table is used to store the Interest access information. When an Interest arrives, the AIT will record the request and router hops. It should be note that each caching node can obtain the router hops from itself to the data consumer based on the information exchanged by the OSPFN protocol [25]. We list all the fields of AIT in the Table 1. The number of future data requests can be predicted based on the data in the AIT. The prediction result can be approximated as user access frequency.

Table 1. Access Information Table

Field	Data Type	Meaning
Name	String	Use data name as a unique identifier
Frequency	Float	The number of Interest in fixed time period
Time	time	Fixed time period of this data
Hops	Float	The average router hops from it to the data consumer
StoreFlag	Boolean	Indicates whether the corresponding data is stored

7 Performance Evaluation

In this section, we evaluate the performance of the proposed cache placement strategy through simulations experiments. NDNSIM [6-7] has been chose as the experiment platform.

7.1 Experimental Environment

We use the NDNSIM applications simulation to generate realistic synthetic workloads. The principle of generate is similar with [26-28]，we generate data in a random way. We assume that the total numbers of data type in the network is 10^3. The size of each cache data block is consistent with [29-30]. According to the research in [28], we simulate the popularity of the content by a Zipf law of parameter z_p. In order to approximate the Zipf distribution, we have given five different z_p values, *i.e.* $z_p \in Z = \{0, 0.3, 0.5, 0.7, 1\}$. If $z_p = 0$, it means that every data type has equally popular.

In order to evaluate the performance of the algorithm, we chose two different sets of network topologies. The first set of network topologies is regular topology, e.g. star topology, grid topology and ring topology. The second set of network topologies comes from the Internet Topology Zoo dataset [32]. We assume that each node in the network hosts a cache and it can send data request. And the request rate of each data type at every consumer node is determined by its popularity. For router node, each arrival requests are independent. In all the experiments, we assume that each node generate a total of 200 requests per second. Thus, the request rate of each data type at each node varies from 0 to 200 req/sec (r/s) according to its popularity. At the beginning of the experiment, each cache node is empty by default.

We evaluate the performance of the caching strategies from following aspects:

Network traffic. The overall network traffic cost per second (request*hops/second).

Replacement frequency. The replacement frequency represents the average number of times each node take cache replacement.

Server hit reduction ratio. It reflects the load saving of servers due to the in-network cache hits.

7.2 Comparison of the Lagrangian Heuristic Algorithm and Distributed Cache Algorithm

We compare LHA and DCA through two sets of experiments to confirm the reliability and feasibility. Figure 2 and Figure 3 depict the performance comparison of the proposed two different algorithms, respectively, in the regular topology and Internet Topology Zoo dataset. Since the value of z_p in the real network is between 0.6 and 0.9 [31], we set $z_p = 0.7$ in this experiment.

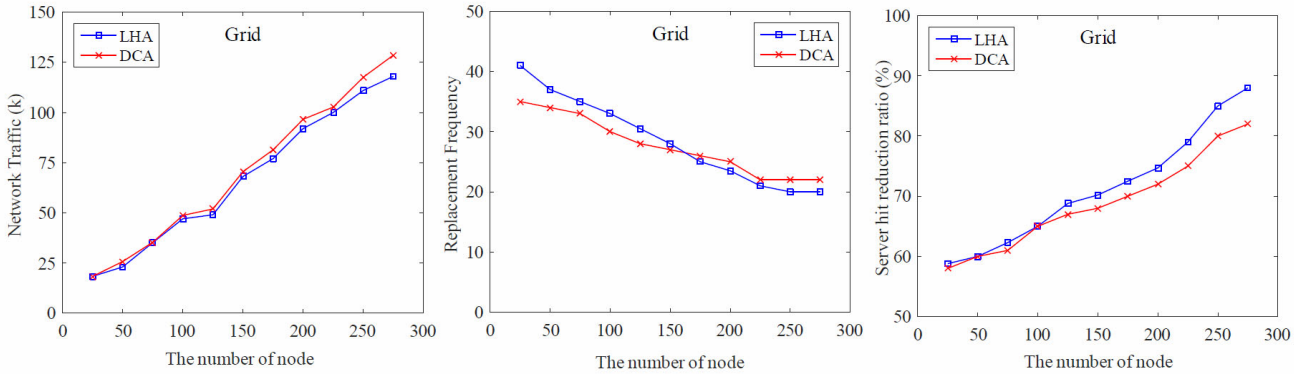


Figure 2. The performance of the proposed LHA and DCA using the regular network topology (grid topology)

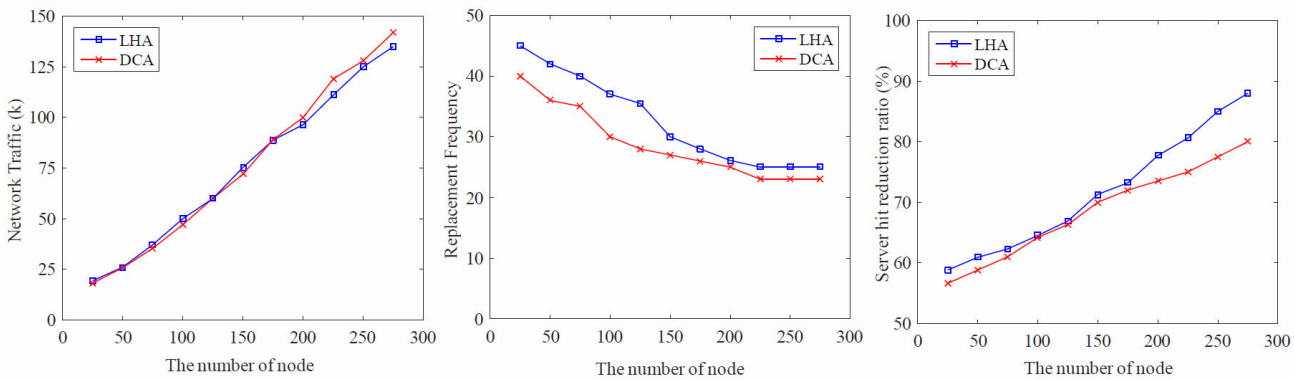


Figure 3. The performance of the proposed LHA and DCA using the Internet Topology Zoo dataset

We observe that when the number of nodes increased from 25 to 275, the network traffic continues to increase and the DCA's network traffic is always more than LHA's. However, the gap between the LHA and DCA did not exceed 11.7%, the average is 4.9%. We have analyzed the contents of the cache at different time points and found that the average similarity between them is 88.7%. And it can be seen in the Figure 2 and Figure 3, the replacement frequency and Server hit reduction ratio are also similar. This means that the Distributed Cache Algorithm's performance is very close to LHA.

7.3 Comparison of the Different Cache Algorithm

In order to evaluate the performance of distributed caching algorithm, we compare it with different kind of cache algorithm, including First Input First Output algorithm (FIFO), Least Recently Used algorithm (LRU), Content Popularity algorithm (CPA) [12] and MAGIC algorithm (MAGIC) [13]. Consider the number of nodes, Figure 4 and Figure 5 depict the performance comparison of different kind of algorithms, respectively, in the regular topology and Internet Topology Zoo dataset. Since the different z_p has implies that the aggregate request rate generated at each node is different, we take experiment to evaluate of algorithm performance under different content popularity. Figure 6 depict the performance comparison of above algorithms in the regular topology (grid topology) when use the $z_p \in Z = \{0, 0.3, 0.5, 0.7, 1\}$ as the independent variable. And Figure 7 and depict the performance comparison of them by using the Internet Topology Zoo dataset.

We observe that the DCA can reduce the network traffic by about 12.6% on average compared to the other four types of algorithms. Among them, compared with the FIFO algorithm, the network traffic is reduced by 37.2% when the number is 275. For the permutation frequency, the fluctuation when using of the Internet Topology Zoo dataset is significantly greater than when using the regular topology. This fluctuation is particularly noticeable for FIFO and content popularity algorithms. It means that the above two algorithms are not suitable for more complex conditions network topology. The experimental results show that the replacement frequency decreased by an average of 22.4%. Server hit reduction ratio is another important indicator to evaluate the algorithm. For LRU and FIFO algorithm, their server hit reduction ratio doesn't increase as the number of nodes increases. And their server hit reduction ratio fluctuates around 55% and 60%, respectively. The server hit reduction ratio of the remaining algorithm increases with the increase of the number of nodes. The highest server hit reduction ratio of CPA, MAGIC, DCA are 74%, 75% and 80%, respectively. The data with high content popularity can't form a stable cache when the number of router nodes is less. With the increase in the number of nodes, most of data can be cached in the router nodes and hence improve the server hit reduction ratio, until they reach the bottleneck.

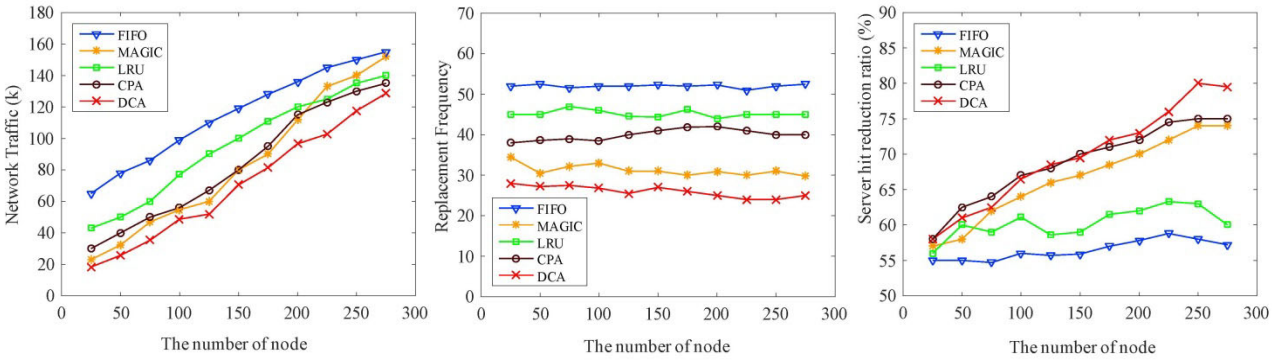


Figure 4. Comparison of performance between different algorithms using the regular topology ($z_p = 0.5$)

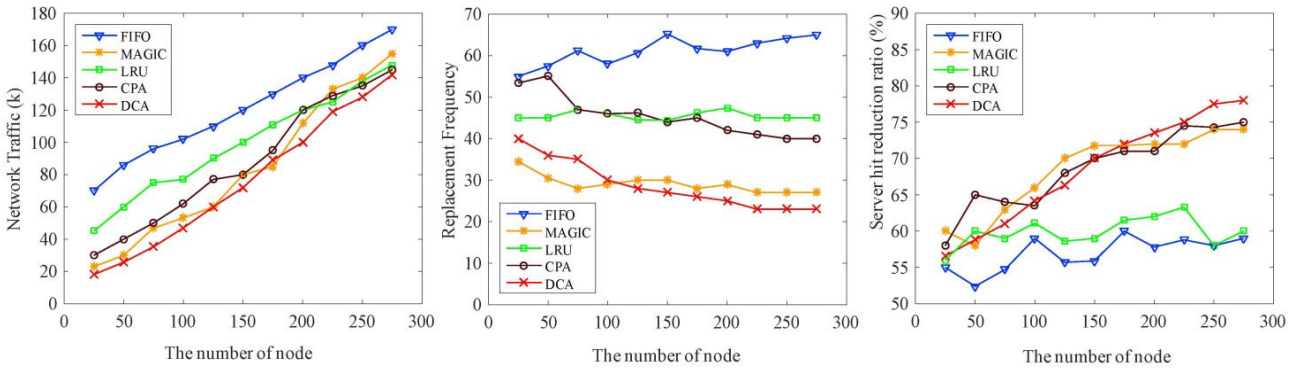


Figure 5. Comparison of performance between different algorithms using the Internet Zoo dataset ($z_p = 0.5$)

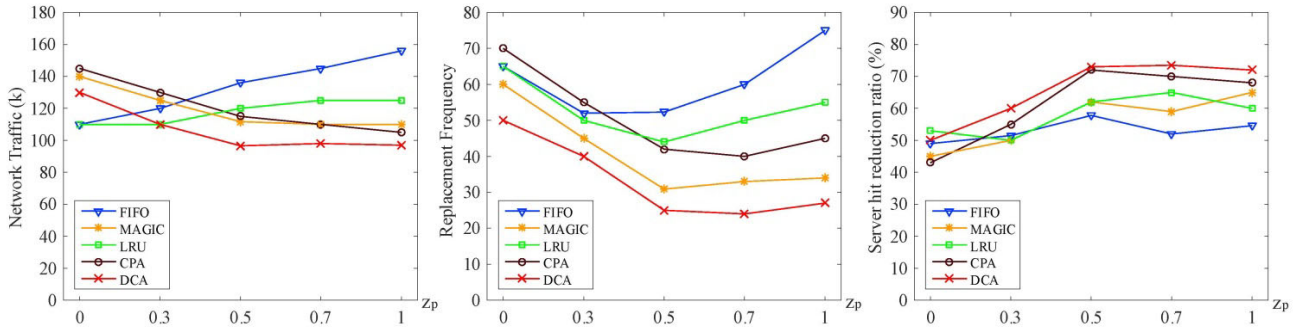


Figure 6. Comparison of performance between different algorithms using the regular topology

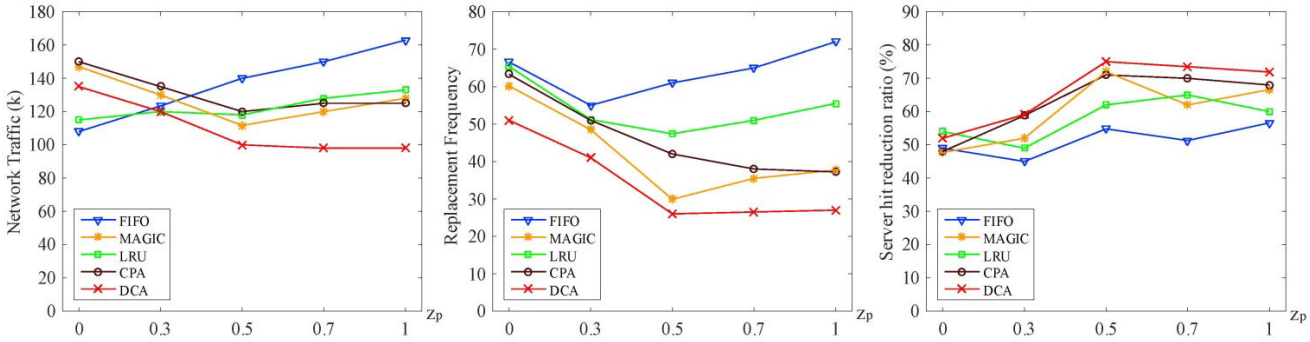


Figure 7. Comparison of performance between different algorithms using the Internet Zoo dataset

As is shown in Figure 6 and Figure 7, we compare the performance of various algorithms under different content popularity conditions. The number of nodes used in this experiment is 200. When the z_p = 0, it means that all kind of data types has the same content popularity. Among them, Figure 6 is shows the experimental results using a regular topology, and Figure 7 shows the experimental results using the Internet Topology Zoo dataset.

We observe that the server hit reduction ratio between 45% and 55%, when z_p = 0. In this case, LRU algorithm and FIFO algorithm's network traffic is lower than our DCA algorithm. However, their replacement frequency is significantly higher than the DCA algorithm. This is due to the fact that the traditional FIFO and LRU algorithms can't determine the optimal initial position of the cache resulting in frequent content replacement. Since the request rate of each content is the same at this time, the algorithm which is based on the content popularity can't determine the hot content. Therefore, they produce higher network traffic. For our DCA algorithm, due to the fact that each data type has the same request rate, the convergence solution can't be obtained. Hence, the network traffic is higher than the FIFO and LRU algorithm. When the value of z_p is higher than 0.3, the performance of our DCA algorithm is significantly higher than other algorithms, regardless of network traffic, replacement frequency or server hit reduction ratio.

Through [28], we know that the value of z_p in the network is between 0.6 and 0.9. Therefore, it can be considered that our algorithm's performance is higher than other algorithms. Among them, the server hit reduction ratio is no more than 75% no matter what kind of z_p. And when the z_p is between 0.5 and 1, the replacement frequency and network traffic are stable for our DCA algorithm.

We do some experiments to choice the best λ value. In Figure 8, we describe the effect of λ on our algorithm in the case of different content popularity. In this experiment, we just use the topology from the Internet Zoo Topology dataset. The results show that when the λ is between 2 and 3, our algorithm achieves the best performance.

Finally, in order to examine the distance (hops) that requests travel in the network until the requested item is found at a cache, we also depict in Figure 9 the performance of FIFO algorithm, CPA algorithm and our distributed caching algorithm. The depicted value is the average percentage of requests for a regular grid topology and a topology from the Internet Zoo Topology dataset. In this experiment, we set the z_p = 0.7 and the order of send data request is random. We observe that the data packets are found closer to the requesting node. More than 80% of the data types are found in less than three hops. Particularly, when using the regular topology, more than 86% of the data types are found in less than three hops. Therefore, the effectiveness of our algorithm is more impressive in the case of the regular topology when z_p = 0.7. However, no matter what kind of topology, our algorithm is better than others.

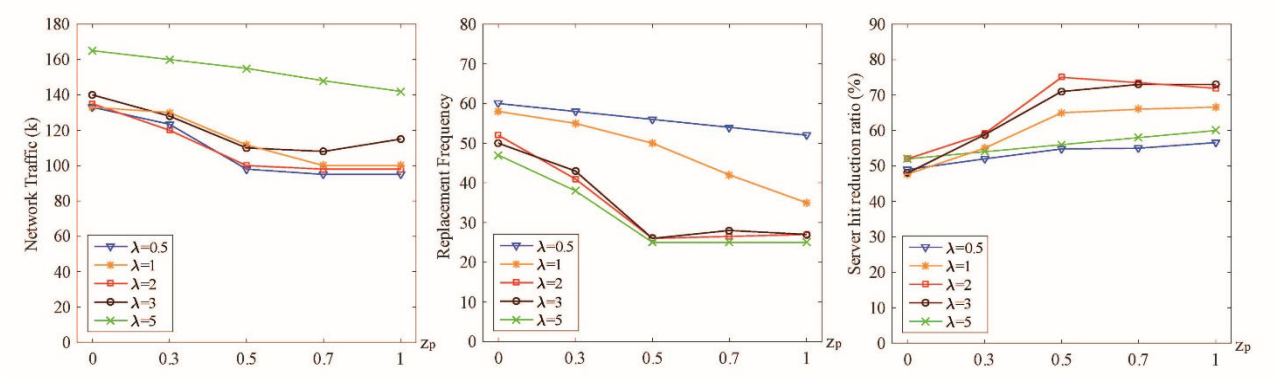
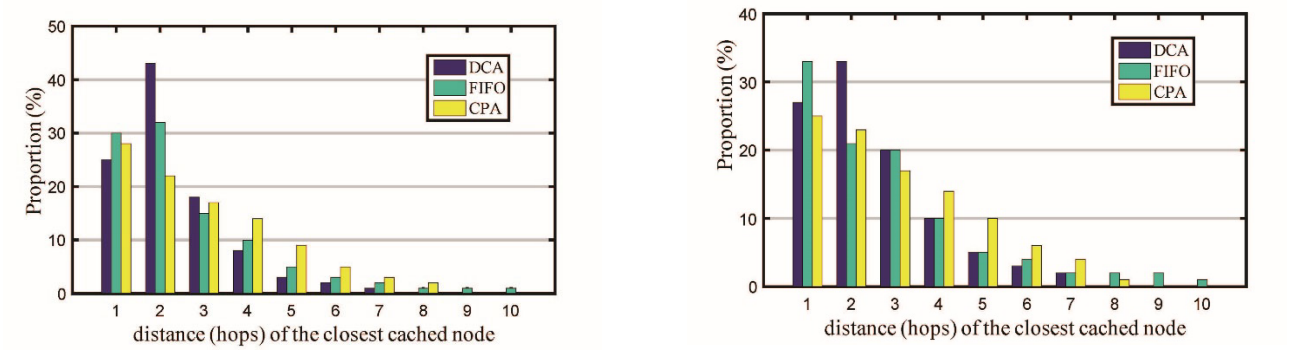


Figure 8. Comparison of performance with different λ (Using the Internet Topology Zoo dataset)



(a) Using the regular topology (z_p = 0.7)

(b) Using the Internet Zoo topology dataset(z_p = 0.7)

Figure 9. The distance (hops) between request node and closest cached node

8 Conclusion

In this paper, we proposed a new cache placement strategy for wireless Internet of Things which can efficiently implement data cache location calculation and cache replacement. In order to solve the problem of optimal cache location selection. First of all, it is transformed into generalized facility location problem (GFLP) by modelling the network. Then, a Lagrangian Heuristic Algorithm (LHA) has been proposed to find the global approximate optimal solution, since the GFLP is an NP-hard problem. Particularly, the Lagrangian relaxation method is used to simplify the calculation process and accelerate the convergence solution. Based on this, we transform the iteration procedure of finding the optimal cache location into local decisions of cache location selection and cache replacement, and we propose a distributed cache placement algorithm. Finally, we implemented the distributed cache placement algorithm in NDN and compared it with other algorithms. The simulation results show that our method is superior to other algorithms in all respects, including network traffic, replacement frequency and server hit reduction ratio.

In particular, we use the regular topology and Internet Zoo dataset as the experiment network topology, since there are no suitable publicly available datasets for our performance evaluation. In this paper, we give the comparison of the algorithm performance when using the two kinds of topologies. The experimental results show that the performance of our algorithm is superior to other algorithms in any topology.

In the future, we will improve the distributed caching algorithm performance by optimize the calculation of local benefit value. Besides, we will also consider the change of the routing interface. In addition, the node resources will also be taken into consideration in order to optimize the algorithm.

Acknowledgments

This work is partly supported by the National Natural Science Foundation of China under Grant 61772085, 61672109, 61532012.

References

[1] D. R. Cheriton, M. Gritter, *TRIAD: A new next-generation Internet architecture*, http://www-dsg. stanford. edu/triad, March, 2000.

[2] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, I. Stoica, A Data-oriented (and beyond) Network Architecture, *ACM SIGCOMM Computer*

Communication Review, Vol. 37, No. 4, pp. 181-192, October, 2007.

[3] G. Zhang, Y. Li, T. Lin, Caching in Information Centric Networking: A Survey, *Computer Networks*, Vol. 57, No. 16, pp. 3128-3141, November, 2013.

[4] L.-Y. Wu, X.-S. Zhang, J.-L. Zhang, Capacitated Facility Location Problem with General Setup Cost, *Computers & Operations Research*, Vol. 33, No. 5, pp. 1226-1241, May, 2006.

[5] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, P. Crowley, E. Yeh, *Named data Networking (ndn) Project*, PARC Tech Report 2010-003, October, 2010.

[6] S. Mastorakis, A. Afanasyev, I. Moiseenko, L. Zhang, *ndnsim 2: An Updated Ndn Simulator for ns-3*, Technical Report NDN-0028, November, 2016.

[7] A. Afanasyev, I. Moiseenko, L. Zhang, *ndnsim: Ndn Simulator for ns-3*, Technical Report NDN-0005, October, 2012.

[8] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, A Survey of Information-centric Networking, *IEEE Communications Magazine*, Vol. 50, No. 7, July, 2012.

[9] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, Cache Coherence in Machine-to-machine Information Centric Networks, *LCN '15 Proceedings of the 2015 IEEE 40th Conference on Local Computer Networks*, Clearwater Beach, FL, 2015, pp. 430-433.

[10] S. Charpinel, C. A. S. Santos, A. B. Vieira, R. Villaca, M. Martinello, Sdccn: A Novel Software Defined Content-centric Networking Approach, *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA),* Crans-Montana, Switzerland, 2016, pp.87-94.

[11] X. Sun, Z. Wang, An Optimized Cache Replacement Algorithm for Information-centric Networks, *2015 IEEE International Conference on Smart City/ SocialCom/SustainCom (SmartCity),* Chengdu, China, 2015, pp.683-688.

[12] B.-H. Zhang, H.-C. Zhou, G.-L. Li, H.-K. Zhang, H.-C. Chao, Least Popularly Used: A Cache Replacement Policy for Information-Centric Networking, *Journal Of Internet Technology*, Vol. 17, No. 1, pp.1-10, January, 2016.

[13] J. Ren, W. Qi, C. Westphal, J. Wang, K. Lu, S. Liu, S. Wang, Magic: A Distributed Max-gain In-network Caching Strategy in Information-centric Networks, *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS),* Toronto, ON, Canada, pp. 470-475.

[14] X.-C. Sun, Z.-J Wang, H, Chu, Q.-R. Zhang, An Efficient Resource Management Algorithm for Information Centric Networks, *Journal Of Internet Technology*, Vol. 17, No. 5, pp.1007-1015, September, 2016.

[15] W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, J. Cai, Content Popularity Prediction and Caching for ICN: A Deep Learning Approach with SDN, *IEEE Access*, December, 2017.

[16] H.-B. Wu, J. Li, J. Zhi, Probability-based Heuristic Content Placement Method for ICN Caching, *Journal on Communications*, Vol. 37, No. 5, pp. 62-72, May, 2016.

[17] Y. Zhang, X.-B. Tan, W.-P. Li, PPC: Popularity Prediction Caching in ICN, *IEEE Communications Letters*, Vol. 22, No. 1, pp. 5-8, January, 2018.

[18] I. Psaras, W. K. Chai, G. Pavlou, In-network Cache Management and Resource Allocation for Information-centric Networks, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 11, pp. 2920-2931, December, 2014.

[19] J. Barceló, J. Casanovas, A Heuristic Lagrangean Algorithm for the Capacitated Plant Location Problem, *European Journal of Operational Research*, Vol. 15, No. 2, pp. 212-226, February, 1984.

[20] G. Ghiani, L. Grandinetti, F. Guerriero, R. Musmanno, A Lagrangean Heuristic for the Plant Location Problem with Multiple Facilities in the Same Site, *Optimization Methods and Software*, Vol. 17, No. 6, pp. 1059-1076, October, 2002.

[21] N. Christofides and J. E. Beasley, Extensions to a Lagrangean Relaxation Approach for the Capacitated Warehouse Location Problem, *European Journal of Operational Research*, Vol. 12, No. 1, pp. 19-28, Jane, 1983.

[22] M. L. Fisher, An Applications Oriented Guide to Lagrangian Relaxation, *Interfaces*, Vol. 15, No. 2, pp. 10-21, April, 1985.

[23] J. E. Beasley, Lagrangean Heuristics for Location Problems, *European Journal of Operational Research*, Vol. 65, No. 3, pp. 383-399, March, 1993.

[24] J. Barcelo, E. Fernandez, K. O. Jörnsten, Computational Results from a New Lagrangean Relaxation Algorithm for the Capacitated Plant Location Problem, *European Journal of Operational Research*, Vol. 53, No. 1, pp. 38-45, July, 1991.

[25] L. Wang, A. Hoque, C. Yi, A. Alyyan, B. Zhang, *Ospfn: An Ospf Based Routing Protocol for Named Data Networking*, Technical Report NDN-0003, July, 2012.

[26] Z.-W. Yan, H.-P. Chiang, Y.-J. Park, X.-D. Lee, Y.-M. Huang, Scalable and Secure Information-centric Networking, *Journal of Internet Technology*, Vol. 14, No. 6, pp. 867-880, November, 2013.

[27] S. Wang, J. Bi, J. Wu, Cache Policy Performance for Informationcentric Networking under a Hop-number-based Metric Framework, *Journal Of Internet Technology*, Vol. 17, No. 3, pp. 409-420, May, 2016.

[28] A. Majumder, N. Shrivastava, R. Rastogi, A. Srinivasan, Scalable Content-based Routing in Pub/Sub Systems, *IEEE International Conference on Computer Communications*, Rio de Janeiro, Brazil, 2009, pp. 567-575.

[29] F. Cao, J. P. Singh, Efficient Event Routing in Content-based Publish-subscribe Service Ntworks, *IEEE International Conference on Computer Communications,* Hong Kong, China, 2004, pp. 929-940.

[30] A. Carzaniga, M. J. Rutherford, A. L. Wolf, A Routing Scheme for Content-based Networking, *IEEE International Conference on Computer Communications*, Hong Kong, China, 2004, pp. 918-928.

[31] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, Web Caching and Zipf-like Distributions: Evidence and Implications, *IEEE International Conference on Computer*

Communications, New York, NY, 1999, pp. 126-134.

[32] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden M. Roughan, The Internet Topology Zoo, *IEEE Journal on Selected Areas in Communications*, Vol. 29, No. 9, pp. 1765-1775, October, 2011.

Biographies

Hua Wei is currently a Ph.D. candidate in Beijing University of Posts and Telecommunication. His main research interests include Internet of Things and Information Center Network.

Hong Luo is a professor of the School of Computer Science, Beijing University of Posts and Telecommunications, China. She is also a research member of the Beijing Key Lab of Intelligent Telecommunication Software and Multimedia. Her research interests include Internet of Things, smart environments, data service and communication software.

Yan Sun is a Professor of the School of Computer Science, Beijing University of Posts and Telecommunications, China. She is also a research member of the Beijing Key Lab of Intelligent Telecommunication Software and Multimedia. Her research interests include Internet of Things, sensor networks, smart environments and embedded systems.