Computer Networks 110 (2016) 266–283

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Design and evaluation of coordinated in-network caching model for content centric networking☆

Yuemei Xu[a,*], Song Ci[b], Yang Li[c], Tao Lin[c], Gang Li[d]

[a] Department of Computer Science, Beijing Foreign Studies University, Beijing, China
[b] Department of Electrical and Computer Engineering, University of Nebraska-Lincoln, NE, USA
[c] Institute of Information Engineering, Chinese Academy of Science, Beijing, China
[d] School of Information Technology, Deakin University, Australia

ARTICLE INFO

Article history:
Received 2 April 2016
Revised 30 September 2016
Accepted 5 October 2016
Available online 6 October 2016

Keywords:
In-network caching
Dynamic request routing
Content centric networking

ABSTRACT

In-network caching has been widely adopted in Content Centric Networking (CCN) to accelerate data delivery, mitigate server load and reduce network traffic. However, the line-speed requirement makes the in-network caching space very limited. With the rapid growth of network traffic, it is significant challenging to decide content placement in such limited cache space. To conquer this conflict, coordinated in-network caching schemes are needed so as to maximize the profit of ubiquitous caching capacities. In particular, in-network caching in CCN is deployed as an arbitrary network topology and naturally supports dynamic request routing. Therefore, content placement scheme and dynamic request routing are tightly coupled and should be addressed together. In this paper, we propose a coordinated in-network caching model to decide the optimal content placement and the shortest request routing path under constraints of cache space and link bandwidth in a systematic fashion. Via extensive simulations, the effectiveness and efficiency of our proposed model has been validated.

1. Introduction

As an emerging predominant next-generation network architecture, Content Centric Networking (CCN) [1] leverages ubiquitous `in-network caching`, where each router is equipped with a content store (CS) module, to accelerate data delivery, mitigate server load and improve user experience. Furthermore, CCN identifies each piece of content chunk with a globally unique name. As these name-based content chunk can be addressed, stored and transported independently, in-network caching allows every cache enable node (e.g., router, proxy cache and end-user machine) of the network to opportunistically cache the named-chunks and use them to serve the subsequent requests, avoiding going all the way to the original server.

Despite many benefits are expected by in-network caching, there are a number of challenges to be addressed. First, with current technologies, line-speed requirement limits the capacity of in-network caching and suggests the cache size of each router should not exceed 10GB DRAM memory size [2]. It is significant challenging to decide content placement in such limited cache space. Second, the increasing demand of network traffic, such as user-generated content, time-shift TV and high definition video, overwhelms link bandwidth capacities. The scarcity of bandwidth of Internet links is becoming a bottleneck of accessing huge amount of data. We therefore argue that designing an effective in-network caching scheme in consideration of cache space and link bandwidth constraints is critical for CCN. This problem has been studied extensively in traditional Web caching or content distribution network (CDN) systems, which, however, are not fit for CCN due to the following two reasons.

First, unlike content caching deployed as an *overlay* capacity in CDN, in-network caching is deployed as an *underlay* network capacity in CCN. Therefore, caching nodes in CCN may cover the whole Internet scale and are arranged as an *arbitrary* network topology (see Fig. 1(b)), rather than the hierarchical tree structure in CDN (see Fig. 1(a)). Second, in-network caching naturally supports *dynamic request routing*, which can locate content replicas based on current caching statuses in the network and selects a more benefit path for the request. Fig. 1(b)) illustrates an example of dynamic request routing in an arbitrary network topology. When router A receives a request from an end user, dynamic request routing strategy will send it in path 1 or 2 (A-C-B or A-C-D),

☆ This work is supported by the Fundamental Research Funds for the Central Universities of China (No.023600-500110002), the National Natural Science Foundation of China (No.61502038).

* Corresponding author.
E-mail address: xuyuemei@bfsu.edu.cn (Y. Xu).

http://dx.doi.org/10.1016/j.comnet.2016.10.004
1389-1286/

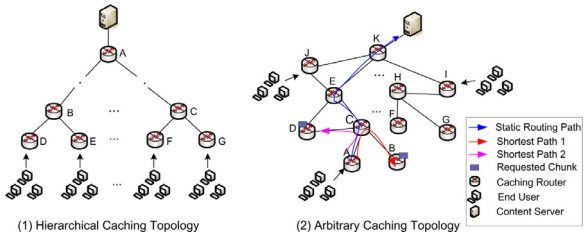


Fig. 1. Hierarchical and arbitrary caching network topologies comparison, and static and dynamic request routing paths comparison in CCN.

both of which are much better than the one (A-C-E-K-Server) selected by static routing strategy, sending the request to the original server.

Existing works [2–7] have been conducted on designing in-network caching strategies in CCN, such as the optimal content placement locations [3], explicit cooperative caching scheme[4], implicit cooperative caching scheme[5], name-based request routing strategy [6], bandwidth and storage sharing mechanism [7] and cache networks modeling [8]. However, very limited works focus on designing in-network caching schemes under both constraints of cache space and link bandwidth. Furthermore, prior researches in CCN usually rely on an opportunistic cache-hit (happen-to-meet) fashion to utilize the in-network storages. In the happen-to-meet fashion, requests are forwarded using static request routing strategy as that in CDN. Therefore, only the content cached on the path towards the original server can be utilized, which limits the network-wide usage of the in-network caching storages. To maximize the profit of ubiquitous in-network caching capacities in CCN, dynamic request routing should be adopted and be coupled with content placement location. That is the content placement status affects how to select an appropriate dynamic request routing path, while the request routing path is the reverse of content delivery path in CCN, therefore potentially decides which routers in the network will store the content.

To address the aforementioned issues, we focus on the in-network caching scheme of CCN under constraints of cache space and link bandwidth, and propose a coordinated in-network caching model to capture the characteristics of *arbitrary network topology* and *dynamic request routing requirement* of CCN. In particular, two important and coupled problems are carefully addressed in the model: (1) **content placement problem.** CCN routers decide whether to cache the receiving named-data or not to make maximum benefit of their limited cache space, which is defined as content placement problem. (2) **dynamic request routing problem.** Routers exploit network-wide in-network storage and forward requests to the nearby content replicas. The optimal request routing path dynamically changes due to the high volatility of data in caches and it takes efforts to find an appropriate (e.g. shortest) path to forward requests, which is defined as dynamic request routing problem. Our objective is to minimize the aggregate bandwidth cost of data transfer in the network through solving the model to find the optimal content placement decision and dynamic request routing path. The main contributions of this paper are as follows:

- We propose a complete model to capture the characteristics of *arbitrary network topology* and *dynamic request routing requirement* of CCN and then formulates the coupled content placement and dynamic request routing problem under link bandwidth and cache space constraints.
- By adopting the Lagrangian dual technique, the coupled content placement and request routing problem can be divided into two subproblems, each of which is the problem of maximizing a submodular function subject to a matroid constraint therefore can be solved with a greedy algorithm, achieving at least 1/2 of the optimal solution in terms of bandwidth cost savings.
- Extensive experiments are carried out to evaluate the performance of the proposed model and show that the proposed approach is superior compared with the existing caching schemes.

The rest of this paper is organized as follows. Section 2 reviews a survey of the related work. Section 3 presents our coordinated in-network caching model. Problem formulation and solutions are given in Section 4. Section 5 shows the experiment settings and the experimental results for verifying the effectiveness of our approach. Section 6 summarizes the observations and findings in experiments. Finally, Section 7 concludes the paper.

2. Related work

Existing works on jointly coupled content placement and request routing problem can mainly classified into two categories: (i) for traditional Web caching systems and CDN; (ii) for in-network caching of CCN.

As to caching for traditional Web caching system and CDN, several researches [9–11] have been carried out to perform *centralized* content placement and request routing in networks with *hierarchical topologies*. Borst et al. [9] presented a collaborative content placement and requesting routing strategy for a two-layer hierarchical caching topology in CDN. It solved the collaborative caching problem in CDN using a linear optimization technique. Dai et al. [10] proposed several mechanisms for the cooperation between different layers in a hierarchical IPTV topology for the purpose of maximizing the overall traffic. Jiang et al. [11] considered groups of “last mile” CDN servers, e.g., set-top boxes located within users’ homes, and proposed a set of mechanisms to jointly manage content replication and request routing within this architecture. All these works show promising direction to exploit the coupled problem of content placement and request routing. However, request routing in these studies generally represents sever selections strategies in CDN. Our works focuses on dynamic request routing upon ubiquitous in-network caching routers. Furthermore, these works were designed for hierarchical network topology in the context of CDN or IPTV and can not apply to the arbitrary network topology of CCN.

With respect to in-network caching for CCN, some researches have also conducted on the joint problem of content placement and request routing in CCN. Saino et al. [12] proposed a hash-function-based content placement and request routing process. It used a hash function to map named-content identifiers to cache nodes and then designed five hash-routing schemes with prior knowledge of content placement locations. This approach was similar to the centralized content placement in CDN but adopted facility request routing mechanism, and may be more suitable in small-scale caching topology. Some works introduced to use request forwarding information to improve the efficiency of content placement. For example, Kawano et al. [13] proposed a selective content placement strategy based on the request history in CCN, which suggested routers not to cache the chunk less likely to be used next according to the historical information of requests. Wang et al. [14] designed a collaborative caching scheme by partitioning content space and hash-routing.

Besides, existing works also exploited to use content placement information to benefit request forwarding strategies in CCN. Li et al. [15] proposed a CLS scheme which encouraged routers to maintain content delivery trails information and used them to direct requests. Eum et al. [16] proposed a secondary best-effort request routing mechanism name potential based routing (PBR) to boost availability of copies. Chiocchetti et al. [17] used a reinforcement learning to discover available content replicas and designed

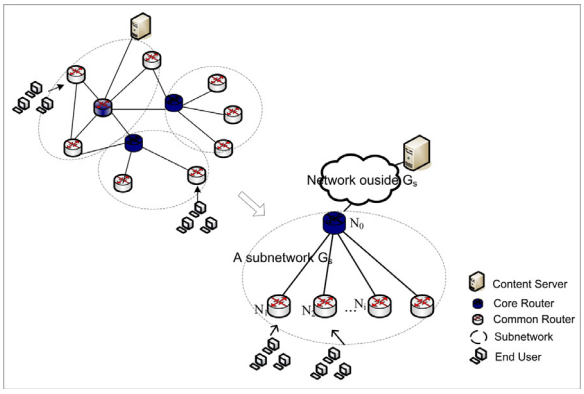


Fig. 2. An example of network decomposition and graphical illustration of a subnetwork.

a dynamic request forwarding scheme, called INFORM. Wang et al. [18] designed a dynamic request forwarding strategy, which suggested n-hop neighboring routers to exchange cache status information and thus could realize content-ware request forwarding, but might cause considerable information exchange overhead. Yi et al. [19] investigated an adaptive forwarding strategy in CCN. In [19], available interfaces were periodically probed to retrieve content via the best path. All these works [15]-[19] leveraged content placement information to find request routing path in CCN. However, the determination of request routing in these works is decoupled from content placement decisions, which assumes the content placement decisions have been made and focuses on finding a nearby copy of content based on current cache status. Actually, the request routing path in CCN will greatly affect the content placement location, as it is the reverse of content delivery path. The request routing and content placement problems are coupled tightly and should be solved together to maximize the in-network caching profit of CCN. Furthermore, link bandwidth constraint and heterogeneous link bandwidth capacity are neglected by these works. A more effective strategy to achieve coordinated content placement and request routing in CCN, under consideration of link bandwidth constraint and arbitrary caching network topology, still remains a great challenge.

Our work differs from above existing works in three-fold: (1) we consider an architecture of arbitrary in-network caching topology with limited link bandwidth capacity. (2) a comprehensive formulation of coupled content placement and request routing problem in CCN is presented, aiming to maximizing the bandwidth cost savings. (3) Lagrangian dual technique is adopted to solve the proposed formulation and a dynamic request routing mechanism combined with optimal content placement is designed, achieving a good tradeoff between considerable bandwidth cost savings and incurred content access delay.

3. In-network caching model

In this section, we present our coordinated in-network caching model as shown in Fig. 2. We first describe the decomposition of arbitrary network topology of CCN and then present the model framework.

3.1. Decomposition of arbitrary network topology

The in-network caching routers in CCN compose of an arbitrary network topology. It is unfeasible to address all temporary copies of content in such a large scale arbitrary topology and is time consuming to make optimal content placement for the whole network routers in a distributed manner. Therefore, previously we introduced the concept of dominating set to classify the routers of CCN into core routers and common routers, and further decomposed the arbitrary network topology into subnetworks [20]. The node classification and network decomposition can be referred to our previous work, which are not the focus of this paper.

Fig. 2 shows an example of network decomposition. Black nodes in Fig. 2 represent the core routers and white nodes denote the common routers, which are distinguished by a connected dominating set (CDS) construction algorithm in [20]. For a given network topology and the identified CDS algorithm, subnetwork decomposition is artificially carried out off-line by considering both realistic requirements and topology information. After decomposition, each subnetwork consists of a core router and M common routers that directly connect to the core router, $M \geq 0$. Note that when $M = 0$, it means that a subnetwork may only include a single core router. It such a case, there is no coordination in a subnetwork.

3.2. Model description

A subnetwork G_s consists of a core router and M common routers. Denote all the routers in G_s with set $\mathcal{M} = \{0, 1, 2, \ldots, M\}$, where N_0 is the core router and N_i denotes a common router, $i = 1, 2, \ldots, M$. The routers are endowed with caches of size $B_0, B_1, \ldots, B_M$. Each content item in CCN is segmented into chunks and users send chunk-level requests for a desired content. Repository chunk set $\mathcal{K}$ is a collection of chunks with size s, $K = |\mathcal{K}|$.

All routers have three entities: Forwarding Information Base (FIB), Content Store (CS) and Pending Interest Table (PIT). FIB is used to forward requests toward potential source(s), which is generated based on the CCN name routing protocol. CS caches the content passed by it according to the content placement strategy. PIT keeps track of the requests forwarded upstream toward source(s) so that the returned content can be sent back the its original requester(s). Upon receiving a request, a router will first check whether its local CS can satisfy the request. If yes, traffic can be saved for fetching it from source(s). If not, it will further check whether its PIT has recorded an ongoing request asking for the same content. If there is a matched record in PIT, request forwarding is ended by adding a new request incoming interface to the PIT record. If the request cannot be hit by CS or PIT, common router will send it to the core router in G_s. If core router's CS or PIT still cannot satisfy the request, it will implement a *dynamic request forwarding*: sending the request downstream to a common router which holds a copy of the requested chunk or sending the request to the source(s) according to FIB. Through *dynamic request forwarding*, core router can coordinate all routers in G_s and saves traffic volume for fetching chunk from source(s).

Common routers receive requests directly from end-users. Denote λ_{ik} as the request rate of chunk k at common router N_i. Core router receive requests either from its neighboring routers outside G_s, denoted by λ_{0k}, or from its directly connected common routers in G_s. Let a $0-1$ decision variable x_{ik} indicate whether chunk k is stored in N_i or not, $i \in \mathcal{M}$.

Denote by c_0 the unit bandwidth cost of transferring chunk from server to core router, by c_i the unit cost associated with transferring chunk between core router and common router N_i, $i = 1, \ldots, M$. Then $c_i + c_j$ is the unit cost of transferring chunk between common routers N_i and N_j. We assume that $c_0 + c_i > c_i + c_j$, because it costs less to fetch a chunk from a peer common router in G_s than from the original server.

The links in G_s have heterogeneous bandwidth capacities. Let U_i be the available uplink bandwidth of common router N_i for serving the requests from other common routers. The link capacity between a pair of directly connected common routers is represented by $U_{ii'}$. Table 1 summarizes the key notations in this paper.

Table 1
Summary of key notations.

Notation	Meaning
G_s	Subnetwork.
$\mathcal{M}$	Set of routers in G_s.
$\mathcal{K}$	Set of chunks.
s	Size of chunk k.
N_i	Router in G_s, $i \in \mathcal{M}$.
λ_{ik}	Request rate of chunk k at N_i.
B_i	Storage capacity of N_i.
x_{ik}	Indicator for caching chunk k at N_i.
c_0	Unit cost of sending a chunk from server to N_0.
c_i	Unit cost of sending a chunk between core router and common router N_i.
$c_{ii'}$	Unit cost of sending a chunk between a directly connected common router N_i and $N_{i'}$.
U_i	Available uplink bandwidth of common router N_i.
$U_{ii'}$	Link bandwidth between a pair of directly connected common router N_i and $N_{i'}$.
q_{ik}	The proportion of the uplink bandwidth U_i is dedicated to serve chunk k.

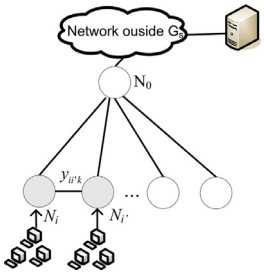


Fig. 3. Interconnected common router coordination.

4. Designing coordinated in-network caching schemes

Our model explores router coordination in G_s, which includes coupled operations of *optimal content placement* and *dynamic request routing*, aiming at minimizing the total bandwidth cost subject to cache space and link bandwidth constraints. This can be transferred to an equivalent problem of maximizing the bandwidth cost savings subject to cache space of routers and link bandwidth constraints.

Common routers may be connected directly or via the core router. Therefore, the coordination among interconnected common routers is first discussed and the coordination among common routers connected via the core router is then investigated.

4.1. Interconnected common router coordination

As denoted in Fig 3, fully connected common routers are able to disseminate chunks via direct links. If a request for chunk k is satisfied at common router N_i, the unit bandwidth cost saving is $c_0 + c_i$. If the request in N_i is fulfilled by a directly connected common router $N_{i'}$, the unit bandwidth cost saving is $c_0 + c_i - c_{ii'}$. The notation $y_{ii'k}$ indicates whether requests for chunk k at router N_i are served from $N_{i'}$ or not. Then the cost savings for interconnected common router coordination can be calculated as follows:

$$\text{Maximize} \sum_{i\in\mathcal{M}\setminus\{0\}}\sum_{k\in\mathcal{K}}\lambda_{ik}s[x_{ik}(c_0+c_i) + \sum_{i'\in\mathcal{M}\setminus\{i\}} y_{ii'k}(c_0+c_i-c_{ii'})] \quad (1)$$

$$\text{Subject to} \sum_{k\in\mathcal{K}} x_{ik}s \le B_i, \forall i \in \mathcal{M}\setminus\{0\}. \quad (2)$$

$$x_{ik} + \sum_{i'\in\mathcal{M}\setminus\{i\}} y_{ii'k} \le 1, \forall i \in \mathcal{M}\setminus\{0\}, k \in \mathcal{K}. \quad (3)$$

$$y_{ii'k} \le x_{i'k}, \ \forall i, i' \in \mathcal{M}\setminus\{0\}, k \in \mathcal{K}. \quad (4)$$

Constraint (2) is the storage capacity constraint. Constraint (3) depicts the relation between chunk placement and request forwarding coordination among common routers. The proportion of requests satisfied by local storage or by interconnected common routers should not exceed 1. Constraint (4) limits that request of chunk k at N_i can be forwarded to $N_{i'}$ only when $N_{i'}$ has cached the requested chunk. In practice, we observe that the directly connected common routers are usually placed nearly while their interconnecting links are typically with high bandwidth capacities. Assume that the link capacity $U_{ii'}$ is consistently larger than $\sum_{k\in\mathcal{K}}\lambda_{ik}s$, $i \in \mathcal{M}$ and the unit cost $c_{ii'}$ can be omitted. The leads to the following theorem, which can be used for coordination among interconnected common routers.

Theorem 1. *Given* $Ks \ge \sum_{i\in\mathcal{M}} B_i$ *and* $U_{ii'} \ge \sum_{k\in\mathcal{K}}\lambda_{ik}s$, *for any chunk* k *such that* $x_{ik} = 1$, *we have* $x_{i'k} = 0$, $\forall\ i, i' \in \mathcal{M}\setminus\{0\}$, $i \ne i'$.

Proof. Suppose in the optimal solution of problems (1)–(4), there exists $x_{ik} = x_{i'k} = 1$, $i \ne i'$. With the assumption $U_{ii'} \ge \sum_{k\in\mathcal{K}}\lambda_{ik}s$ and $c_{ii'} = 0$, we change the value of $x_{i'k}$ from 1 to 0 and the request forwarding decision $y_{i'ik}$ from 0 to 1, which achieves the same cost savings with the optimal solution. Given $Ks \ge \sum_{i\in\mathcal{M}} B_i$, there exist a certain chunk k' with $x_{ik'} = x_{i'k'} = 0$. We set $x_{i'k'}$ from 0 to 1 and the routing decision is made as $y_{ii'k'} = 1$, which achieves an assignment that is better than the optimal solution. This is a contradiction.

Theorem 1 suggests that with high link bandwidth capacity, the interconnected common routers should not cache duplicated chunks. Consequently, in subsequent analysis, the interconnected common routers in G_s are treated as one single common router with aggregated storage capacities, exogenous requests and bandwidth capacities. In the coordination among interconnected common routers, for the local unsatisfied requests, routers can easily route them to their neighboring common routers via direct links, otherwise send them to the core router in G_s. Then the coordination between indirectly connected common routers will be implemented, which will be discussed next. □

4.2. Indirectly connected common router coordination

To maximize the bandwidth cost savings, we now proceed coordination of *optimal content placement* and *dynamic request routing* among indirectly connected common routers, as shown in Fig. 4.

If a request of chunk k is satisfied by local router N_i or by the interconnected common router of N_i, the unit bandwidth cost saving is $c_0 + c_i$. If the request is satisfied by core router N_0, the unit

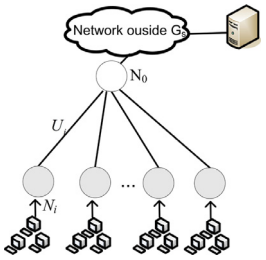


Fig. 4. Indirectly connected common router coordination.

bandwidth cost saving is c_0. If the request is satisfied through *dynamic request routing*, fulfilled by another common router N_j in G_s, the unit bandwidth cost saving is $c_0 + c_i - (c_i + c_j)$. We denote $q_{jk} \in [0, 1]$ as the proportion of uplink bandwidth of common router N_j that is allocated to requests of chunk k, $j \in \{1, \ldots, M\}$. The *dynamic request routing* decision is made at core router based on the value of q_{jk}. Otherwise, the request will be directed to original server without cost savings. The total cost savings caused by coupled operations of *optimal content placement* and *dynamic request routing* can be calculated as follows:

$$\begin{aligned}\text{Maximize} \quad & \sum_{i\in\mathcal{M}\setminus\{0\}}\sum_{k\in\mathcal{K}}\lambda_{ik}s_k(c_0+c_i)x_{ik} \\ & +\sum_{k\in\mathcal{K}}\Big(\sum_{i\in\mathcal{M}\setminus\{0\}}\lambda_{ik}(1-x_{ik})+\lambda_{0k}\Big)s_kc_0x_{0k} \\ & +\sum_{i\in\mathcal{M}\setminus\{0\}}\sum_{k\in\mathcal{K}}q_{ik}U_i(c_0-c_i).\end{aligned} \tag{5}$$

$$\text{Subject to} \quad \sum_{k\in\mathcal{K}}x_{ik}s_k \le B_i, \quad \forall i \in \mathcal{M}. \tag{6}$$

$$\sum_{i\in\mathcal{M}\setminus\{0\}} q_{ik}U_i \le s_k\Psi_{0k}(1-x_{0k}), \quad \forall k \in \mathcal{K}. \tag{7}$$

$$0 \le q_{ik} \le x_{ik}, \quad \forall i \in \mathcal{M}\setminus\{0\}, k \in \mathcal{K}. \tag{8}$$

$$\sum_{k\in\mathcal{K}} q_{ik} \le 1, \quad \forall i \in \mathcal{M}\setminus\{0\}. \tag{9}$$

Constraint (6) is the storage capacity constraint. In constraint (7), $\Psi_{0k} = (\sum_{i\in\mathcal{M}\setminus\{0\}}\lambda_{ik}(1-x_{ik}) + \lambda_{0k})$ is the aggregated incoming request rate in core router. Therefore, $s_k\Psi_{0k}(1-x_{0k})$ represents the total traffic of unsatisfied requests in core router. These requests will be forwarded by *dynamic request routing* decision or be sent to the original server. $\sum_{i\in\mathcal{M}} q_{ik}U_i$ denotes the traffic savings due to *dynamic request routing*, which should not exceed $s_k\Psi_{0k}(1-x_{0k})$. Constraint (8) limits that common routers only reserve uplink bandwidth for the local stored chunks. Constraint (9) implies that the proportion of uplink bandwidth reservation for all chunks should not exceed 1.

The formulated problem in (5)–(9) is a NP-hard problem, which can be proved by restricting (5). Suppose that G_s consists of only one router. The formulated problem is then reduced to a typical Knapsack Problem, which is proved to be NP-hard [21]. Among the existing approaches developed for NP-hard problem, a decomposition based approach is a conventional method. In the following, we first apply Lagrangian relaxation to decompose the formulation into two smaller and easier-to-solve subproblems and further prove each subproblem as a submodular function subject to a matroid constraint.

Constraints (7) and (8) can be incorporated into the objective function by associating a Lagrangian multiplier with each constraint. Constraint (7) associates with η_k and constraint (8) associates with ξ_{ik}. Then the Lagrangian dual problem is represented as:

$$\begin{aligned}&\text{Maximize} \quad \mathcal{L}(\eta_k, \xi_{ik}, x_{ik}, q_{ik}) \\ &\text{Subject to} \quad \sum_{k\in\mathcal{K}} x_{ik}s_k \le B_i, \quad \forall i \in \mathcal{M}. \\ &\qquad\qquad \sum_{k\in\mathcal{K}} q_{ik} \le 1, \quad \forall i \in \mathcal{M}\setminus\{0\}. \\ &\qquad\qquad \eta_k \ge 0, \xi_{ik} \ge 0, \quad \forall i \in \mathcal{M}\setminus\{0\}, k \in \mathcal{K}.\end{aligned} \tag{10}$$

The objective function $\mathcal{L}(\eta_k, \xi_{ik}, x_{ik}, q_{ik})$ is formulated as:

$$\begin{aligned}&\mathcal{L}(\eta_k, \xi_{ik}, x_{ik}, q_{ik}) \\ &= \sum_{i\in\mathcal{M}\setminus\{0\}}\sum_{k\in\mathcal{K}}(\lambda_{ik}s_k(c_0+c_i)+\xi_{ik})x_{ik} \\ &\quad +\sum_{k\in\mathcal{K}} s\Psi_{0k}(\eta_k+(c_0-\eta_k)x_{0k}) \\ &\quad +\sum_{i\in\mathcal{M}\setminus\{0\}}\sum_{k\in\mathcal{K}}(U_i(c_0-c_i-\eta_k)-\xi_{ik})q_{ik}\end{aligned} \tag{11}$$

We now try to decompose the join optimization problem into two subproblems: (1) the *content placement subproblem*; and (2) the *dynamic request routing subproblem*. Through analysis of objective function (11), ξ_{ik} can be omitted by putting constraint (8) into the optimization problem. η_k is decomposed into coefficients η_{ik} to achieve a finer granularity in controlling the cache storage and uplink bandwidth allocation decisions. Then the decomposed two subproblems are as follows:

Subproblem (I):

$$\begin{aligned}&\text{Maximize} \sum_{i\in\mathcal{M}}\sum_{k\in\mathcal{K}}\lambda_{ik}s[\eta_{ik}+(c_0-\eta_{ik})x_{0k}] \\ &\quad +\sum_{i\in\mathcal{M}\setminus\{0\}}\sum_{k\in\mathcal{K}}\lambda_{ik}s_k[(c_0+c_i-\eta_{ik})+(c_0-\eta_{ik})x_{0k}]x_{ik} \\ &\text{Subject to} \sum_{k\in\mathcal{K}}x_{ik}s_k \le B_i, \quad \forall i \in \mathcal{M}.\end{aligned} \tag{12}$$

Subproblem (II):

$$\begin{aligned}&\text{Maximize} \quad \sum_{i\in\mathcal{M}\setminus\{0\}}\sum_{k\in\mathcal{K}}(c_0-c_i-\eta_{ik})U_iq_{ik} \\ &\text{Subject to} \quad \sum_{k\in\mathcal{K}} q_{ik} \le 1, \quad \forall i \in \mathcal{M}\setminus\{0\}. \\ &\qquad\qquad 0 \le q_{ik} \le x_{ik}, \quad \forall i \in \mathcal{M}\setminus\{0\}, k \in \mathcal{K}.\end{aligned} \tag{13}$$

where subproblem (I) is the *content placement subproblem* and subproblem (II) is the *dynamic request routing subproblem*. By Lagrangian relaxation method, the optimal solution of the joint optimization problem can be found through multiple iterations to solve subproblems (I) and (II) until the Lagrangian multiplier η_{ik} converge. Our simulation studies show that this approach requires a long time to converge, especially when there exists a large number of chunks. Thus we try to relax the time granularity of iterations and find a less time-consuming algorithm with a near-optimal performance.

Let the continuous time $\mathcal{T}$ be divided into a plurality of time slots (*i.e.*, $0, t, \ldots, nt, \ldots$). Each time slot is an iteration interval. Routers collect request incoming information and infer chunk popularity distribution during each time slot. Based on the collected information and other related parameters, such as link bandwidth, cache size of routers and current η_{ik} value, subproblems (I) and (II) will be solved periodically at the end of each time slot. According to solutions of (I) and (II), content placement decisions

are made and the corresponding uplink bandwidth are reserved. As core router keeps track of solutions of (I) and (II), it can realize dynamic request routing to coordinate all routers in G_s. Meanwhile, the Lagrangian multiplier η_{ik}, which controls the content placement and dynamic request routing decisions, will be updated at the end of each time slot for next iteration as:

$$\eta_{ik}(n+1) = \eta_{ik}(n) + \theta(n)(1 - \sum_{k\in K} q_{ik}(n)) \times \left[\sum_{i\in\mathcal{M}} q_{ik}(n)U_i - s_k\Psi_{0k}(n)(1 - x_{0k}(n))\right]$$

where $\eta_{ik}(n)$, $q_{ik}(n)$ and $s_k\Psi_{0k}(n)(1-x_{0k}(n))$ are the values of Lagrangian multiplier, proportion of uplink bandwidth assignment and the desired traffic volume for chunk k during time slot $[(n-1)t, nt]$, respectively. $1 - \sum_{k\in K} q_{ik}(n)$ represents the remaining uplink bandwidth of current allocation and $\theta(n)$ denotes the step size. The insufficient reservation of uplink bandwidth for certain chunk k (i.e., $\sum_{i\in\mathcal{M}} q_{ik}(n)U_i \le s_k\Psi_{0k}(n)(1-x_{0k}(n))$) will lead to a decreasing value of η_{ik}, which then will inspire routers to store and reserve uplink bandwidth for chunk k in the next iteration.

In the followings, we first show the solutions of subproblems (I) and (II), and then present the total coordinated in-network caching algorithms in G_s.

4.2.1. Content placement subproblem

subproblem (I) can be written as a maximization of a submodular function subject to matroid constraints.

Definition 1. A matroid $\mathcal{T}$ is a tuple $\mathcal{T} = (\mathcal{S}, \mathcal{L})$, where $\mathcal{S}$ is a finite ground set and $\mathcal{L} \subseteq 2^{\mathcal{S}}$ (the power set of $\mathcal{S}$) is a collection of independent sets, such that:

1. $\mathcal{L}$ is nonempty, in particular, $\emptyset \in \mathcal{L}$.
2. $\mathcal{L}$ is downward closed; *i.e.*, if $B \in \mathcal{L}$ and $A \subseteq B$, then $A \in \mathcal{L}$.
3. If $A, B \in \mathcal{L}$, and $|A| < |B|$, then $\exists y \in B\backslash A$ such that $A \cup \{y\} \in \mathcal{L}$.

A matroid is a structure that captures and generalizes the notion of linear independent in vector spaces. One example is the partition matroid. In a partition matroid, the ground set $\mathcal{S}$ is partitioned into (disjoint) sets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_l$ and

$$\mathcal{L} = \{X \cap \mathcal{S} : |X \cap \mathcal{S}_i| \le b_i, \forall i = 0, \ldots, l\} \tag{14}$$

for some given parameters b_0, b_1, ... , b_l.

Theorem 2. *The constraints in subproblem (I) can be written as partition matroid on the ground set $\mathcal{S}$ defined in (14).*

Proof. In the optimization subproblem (I), we want to find the optimum way of chunk placement in routers' caches. For subproblem (I), we first define a ground set $\mathcal{S}$. Denoting the placement of chunk k at router N_i by the element f_i^k, the ground set will be:

$$\mathcal{S} = \{f_0^1, f_0^2, \ldots, f_0^K, f_1^1, f_1^2, \ldots, f_M^K\}. \tag{15}$$

The ground set can be partitioned into $M+1$ disjoint sets, $\mathcal{S}_0, \ldots, \mathcal{S}_M$, where $\mathcal{S}_i = \{f_i^1, f_i^2, \ldots, f_i^K\}$ is the set of all chunks that might be placed in the cache space of router N_i. Denote the *placement set* of subproblem (I) as X, $X \subseteq \mathcal{L}$ with

$$X = \{f_i^k : x_{ik} = 1, i \in \mathcal{M}, k \in \mathcal{K}\}$$

Then a set of elements which are placed in the cache of router N_i are equal to $X_i \triangleq X \cap \mathcal{S}_i$. We define a $1 \times N$ boolean vector $\boldsymbol{X_i^b}$ associated to X_i, which indicates the chunk placement of router N_i. If the element f_i^k is in the set X_i, the k-th element of vector $\boldsymbol{X_i^b}$ is 1, otherwise it is 0.

Denote the total size of cached chunks at router N_i as S_X^i, $S_X^i = |X \cap \mathcal{S}_i|$. We can also calculate S_X^i as:

$$S_X^i \triangleq \boldsymbol{X_i^b} \times \begin{bmatrix} s_1 \\ s_2 \\ \cdot \\ \cdot \\ \cdot \\ s_K \end{bmatrix}$$

where $s = s_1 = s_2 = \ldots = s_K$.

Therefore, the constraints on the cache capacity of routers can be expressed as $X \subseteq \mathcal{L}$, where

$$\mathcal{L} = \{X \subseteq \mathcal{S} : |X \cap \mathcal{S}_i| = S_X^i \le B_i, i \in 0, 1, \ldots, M\} \tag{16}$$

Comparing $\mathcal{L}$ in (16) and the definition of the partition matroid in (14), we can see that constraints of subproblem (I) form a partition matroid with $l = M$ and $b_i = B_i$ for $i = 0, 1, \ldots, M$. The partition matroid is denoted by $\mathcal{T} = (\mathcal{S}, \mathcal{L})$. □

Having proved the constraints of subproblem (I) is a matroid, we and substitute $\Psi_{0k} = \sum_i \lambda_{ik} s(1 - x_{ik}) + \lambda_{0k}$ into objective function and rewrite subproblem (I) as following:

Subproblem (I'):

$$\begin{aligned} \text{Max} \quad & \sum_{i\in\mathcal{M}\backslash\{0\}} \boldsymbol{R_i} \left| \begin{matrix} \boldsymbol{X_i^b} \\ \mathbf{1}_{1\times\mathcal{K}} - \boldsymbol{X_i^b} \end{matrix} \right|^{\mathrm{T}} \left| \begin{matrix} c_0 + c_i \\ C_{ik} \end{matrix} \right| + \sum_{k\in\mathcal{K}} \lambda_{0k} s C_{0k} \\ s.t. \quad & X \subseteq \mathcal{L} \\ & X_i = |X \cap \mathcal{S}_i|, \forall x \in \mathcal{M} \end{aligned} \tag{17}$$

where $\boldsymbol{R_i} = (\lambda_{i1}s_1, \lambda_{i2}s_2, \ldots, \lambda_{ik}s, \ldots, \lambda_{i\mathcal{K}}s_{\mathcal{K}})$, $C_{ik} = \eta_{ik} + (c_0 - \eta_{ik})x_{0k}$ and $C_{0k} = \eta_{0k} + (c_0 - \eta_{0k})x_{0k}$.

Definition 2 (Submodular functions)**.** Let $\mathcal{S}$ be a finite ground set. A set function $g : 2^{\mathcal{S}} \to \mathbb{R}$ is submodular if for all set $A \subseteq B \subseteq \mathcal{S}$ and for all $i \in \mathcal{S}\backslash B$:

$$g(A \cup \{i\}) - g(A) \ge g(B \cup \{i\}) - g(B) \tag{18}$$

Theorem 3. *The objective function of subproblem (I') is a monotone submodular function.*

The proof of Theorem 3 will be presented in the Appendix part. A greedy algorithm is a quite common way of maximizing a submodular function subject to a matroid constraint. The greedy algorithm starts with an empty set; at each step, it adds one element with the highest marginal value to the set while maintaining independence of the solution. The precise greedy algorithm is shown in Algorithm 1. Classical results on submodular functions showed that the greedy algorithm achieves $\frac{1}{2}$ of the optimal value [22].

In Algorithm 1, $g(.)$ is the objective function in (17) and $g_X(f) = g(X + f) - g(X)$ is the marginal value of an element f with respect to a placement set X. At every iteration, Algorithm 1 selects the element f_α^β with the highest marginal value. Selecting f_α^β means that the chunk β will be cached by router N_α.

The selected f_α^β element is chosen from all elements in set D. The initial value of set $\mathcal{D}$ is the ground set $\mathcal{S}$. At every iteration, element f_α^β is selected and if router N_α has enough space for chunk β (i.e., $|X_\alpha| + s_\beta \le B_\alpha$), the placement set X_α and X will add f_α^β and at the same time D_α and D will remove f_α^β. If the cache of router N_α becomes full, we remove the entire set D_α, the subset of D which is associated to router N_α. So in the subsequent iterations the elements of router N_α will not be considered. At the end, the greedy algorithm returns the placement set X. The results of placement set X will be used for *dynamic request routing* by the core router in G_s, which will be discussed next.

The computational complexity of Algorithm 1 is decided by multiple iterations from step 3 to step 16. Algorithm 1 decides

Algorithm 1 A greedy chunk placement algorithm.

1: **Initialize:**
$\mathcal{D}_i = \mathcal{S}_i = \{f_i^1, f_i^2, \ldots, f_i^{\mathcal{K}}\}$, $i \in \mathcal{M}$.
$\mathcal{D} = \cup_{i\in M}\mathcal{D}_i$.
$X_i \leftarrow \emptyset$, $i \in \mathcal{M}$.
$X \leftarrow \emptyset$.
2: **while** $\mathcal{D} \neq \emptyset$ **do**
3: $f_\alpha^\beta = \arg\max_{f\in\mathcal{D}} g_X(f)$
4: **if** $g_X(f_\alpha^\beta) = 0$ **then**
5: break;
6: **end if**
7: **if** $|X_\alpha| + s_\beta \le B_\alpha$ **then**
8: $X_\alpha \leftarrow X_\alpha \cup \{f_\alpha^\beta\}$, $X \leftarrow X \cup \{f_\alpha^\beta\}$
9: $\mathcal{D}_\alpha \leftarrow \mathcal{D}_\alpha \backslash \{f_\alpha^\beta\}$, $\mathcal{D} \leftarrow \mathcal{D} \backslash \{f_\alpha^\beta\}$
10: **if** $|X_\alpha| = B_\alpha$ **then**
11: $\mathcal{D} \leftarrow \mathcal{D} \backslash \mathcal{D}_\alpha$
12: **end if**
13: **else**
14: $\mathcal{D}_\alpha \leftarrow \mathcal{D}_\alpha \backslash \{f_\alpha^\beta\}$
15: $\mathcal{D} \leftarrow \mathcal{D} \backslash \{f_\alpha^\beta\}$
16: **end if**
17: **end while**
18: Output X_i, $i \in \mathcal{M}$

the content placement to fulfill the CS of each router. As each iteration decides one content placement, $(M+1)L$ iterations are required, where $M+1$ is the number of routers in G_s, L is the average number of chunks cached by each router. Suppose the time complexity of each iteration is Δ, mainly caused by element selection in a greedy manner. So the complexity of Algorithm 1 is $O(ML\Delta)$, proportional to the number of routers and the average number of chunks cached by each router.

4.2.2. Dynamic request routing subproblem

Here we need solve subproblem (II) and generate the dynamic request routing strategy. Let $u_{ik} = c_0 - c_i - \eta_{ik}$, we can reformulate subproblem (II) as:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i\in\mathcal{M}\backslash\{0\}} \sum_{k\in\mathcal{K}} u_{ik} U_i q_{ik} \\ \text{Subject to} \quad & \sum_{k\in\mathcal{K}} q_{ik} \le 1, \quad \forall i \in \mathcal{M}\backslash\{0\}. \\ & 0 \le q_{ik} \le x_{ik}, \quad \forall i \in \mathcal{M}\backslash\{0\}, k \in \mathcal{K}. \end{aligned} \tag{19}$$

Let Q_k denote the collected traffic requirement of chunk k in core router. Core router is responsible to calculate Q_k as:

$$Q_k = \left(\sum_{i\in\mathcal{M}\backslash\{0\}} \lambda_{ik}(1 - x_{ik}) + \lambda_{0k} \right) (1 - x_{0k}) s_k. \tag{20}$$

In order to find the best path to route Q_k, core router in G_s will generate a routing map $\mathcal{R}$ to implement dynamic request routing, which exploits potential downstream content retrieval among common routers before sending them to the original server. Let $\mathcal{R} = [R_1^1, \ldots, R_1^K; \ldots R_i^k, \ldots, R_i^K; R_M^1, \ldots, R_M^K]$, where R_i^k is the probability of forwarding request of chunk k to common router N_i. $R_i^k > 0$ indicates that common router N_i has reserved uplink bandwidth for requests of chunk k. R_i^k is calculated as

$$R_i^k = \frac{q_{ik} U_i}{Q_k}. \tag{21}$$

In the following, we first determine q_{ik} by solving subproblem (II) and then illustrate generation process of routing map $\mathcal{R}$. Subproblem (II) is a typical Knapsack Problem with M knapsacks, which are the available uplink bandwidth of M common routers. Similarly to subproblem (I), we can represent subproblem (II) as the problem of maximizing a submodular function subject to a matroid constraint.

Let $\mathcal{W}$ be the ground set for uplink bandwidth reservation. Because common routers only reserve uplink bandwidth for their local stored chunks, we have $\mathcal{W} = X$. $\mathcal{W}$ can be partitioned into M disjoint sets: $\mathcal{W} = \{\mathcal{W}_1, \ldots, \mathcal{W}_i, \ldots, \mathcal{W}_M\}$, where $\mathcal{W}_i = X_i$ and X_i is the result of placement set at common router N_i. Denote the uplink reservation set of subproblem (II) as Y, $Y \subseteq \mathcal{W}$:

$$Y = \{f_i^k : x_{ik} = 1, q_{ik} \neq 0, i \in \mathcal{M}\backslash\{0\}, k \in \mathcal{K}\}. \tag{22}$$

If $f_i^k \in Y$, the common router N_i will reserve uplink bandwidth for chunk k. The uplink bandwidth reservation in N_i is represented by $Y_i \triangleq Y \cap \mathcal{W}_i$. We define a $1 \times N$ boolean vector $\boldsymbol{Y_i^b}$ associated to Y_i. If the element f_i^k is in the set Y_i, the k-th element of vector $\boldsymbol{Y_i^b}$ is 1, otherwise it is 0. Denote W_q^i as the total fraction of uplink bandwidth reservation at common router N_i, which is calculated as:

$$W_q^i \triangleq \boldsymbol{Y_i^b} \times \begin{bmatrix} q_{i1} \\ q_{i2} \\ \cdot \\ \cdot \\ \cdot \\ q_{iK} \end{bmatrix}$$

The matroid tuple in subproblem (II) can be expressed as $\mathcal{T}' = (\mathcal{W}, \mathcal{L}')$, where $\mathcal{W}$ is the ground set and $\mathcal{L}'$ is a set of sets subject to the constraints of subproblem (II). $\mathcal{T}'$ is written as:

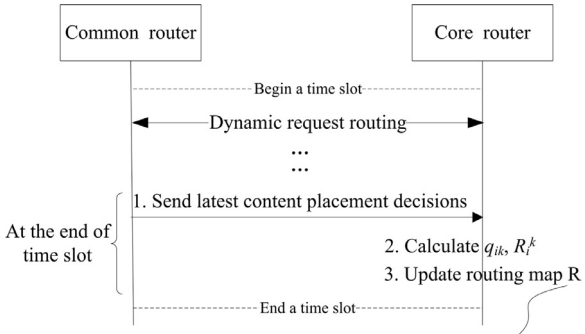
$$\mathcal{L}' = \{Y \subseteq \mathcal{W} : ||W_q^i|| \le 1, i \in 0, 1, \ldots, M\} \tag{23}$$

The objective function of subproblem (II) is also a monotone submodular function and can be proved by similar procedure of subproblem (I). Suppose A' and B' are two uplink bandwidth reservation sets and satisfy $A' \subseteq B' \subseteq \mathcal{W}$. For any $f_i^k \in \mathcal{W}$, subproblem (II) is proved as a monotone submodular objective function by discussing three cases: 1) Chunk k has not been reserved bandwidth at A' and B', e.g., $f_i^k \notin A', B'$. 2) Chunk k has been reserved bandwidth at B' but not at A', e.g., $f_i^k \in B'$, $f_i^k \notin A'$. 3) Chunk k has been reserved bandwidth at A' and B', e.g., $f_i^k \in A', B'$. For these 3 cases, $\mathcal{Z}(A' \cup \{f_i^k\}) - \mathcal{Z}(A') \ge \mathcal{Z}(B' \cup \{f_i^k\}) - \mathcal{Z}(B')$ consistently holds, where $\mathcal{Z}(.)$ is the objective function of subproblem (II) shown at Eq. (19).

Algorithm 2 describes the solution of q_{ik} and the generation of dynamic request routing $\mathcal{R}$. In Algorithm 2, $\mathcal{Z}_Y(f) = \mathcal{Z}(Y + f) - \mathcal{Z}(Y)$ is the marginal value of reserving uplink bandwidth for f with respect to a bandwidth reservation set Y. Initially, $\mathcal{R}$ and Y are both empty. The initial value of $\mathcal{D}$ is the ground set $\mathcal{W}$. At every iteration, Algorithm 2 selects the element f_α^β with the highest marginal value from $\mathcal{D}$. Selecting f_α^β means that common router N_α will reserve uplink bandwidth for chunk β. If the remaining uplink bandwidth of N_α can satisfy the current demand of chunk β (i.e., $U_{temp-\alpha} \ge Q_\beta$), the following operations are implemented (lines 5–9): add f_α^β to the bandwidth reservation set Y_α; reserve $q_{\alpha\beta}$ fraction of bandwidth at N_α for chunk β; add routing probability $R_\alpha^\beta = 1$ to $\mathcal{R}$; update the remain available bandwidth $U_{temp-\alpha}$; remove f_α^β from $\mathcal{D}_\alpha$; and remove $\{f_1^\beta, f_2^\beta \ldots, f_M^\beta\}$ from $\mathcal{D}$, which means that since N_α has reserved bandwidth to satisfy the demand of chunk β, the rest common routers would not reserve bandwidth for it again. Otherwise if the remaining uplink bandwidth of N_α is less than the desired demand of chunk β (i.e., $U_{temp-\alpha} < Q_\beta$), lines 10–13 will be operated. N_α will assign its remain bandwidth to chunk β but the routing probability R_α^β added to $\mathcal{R}$ is less than 1. Since N_α no longer has bandwidth to reserve,

Algorithm 2 Dynamic request routing algorithm.

1: **Initialize:**
$\mathcal{D}_i = \mathcal{W}_i = \{f_i^1, f_i^2, \ldots, f_i^K\}$, $i \in \mathcal{M}\backslash\{0\}$.
$\mathcal{D} = \cup_{i\in\mathcal{M}\backslash\{0\}}\mathcal{D}_i$.
$U_{temp-i} = U_i$.
$Y_i \leftarrow \emptyset$, $i \in \mathcal{M}\backslash\{0\}$.
$\mathcal{R} \leftarrow \emptyset$.
2: **while** $\mathcal{D} \neq \emptyset$ **do**
3: $f_\alpha^\beta = \arg\max_{f\in\mathcal{D}} \mathcal{Z}_Y(f)$
4: **if** $U_{temp-\alpha} \geq Q_\beta$ **then**
5: $Y_\alpha \leftarrow Y_\alpha \cup \{f_\alpha^\beta\}$;
6: $q_{\alpha\beta} = \frac{Q_\beta}{U_\alpha}$;
7: $R_\alpha^\beta = 1$, $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_\alpha^\beta\}$;
8: $U_{temp-\alpha} = U_{temp-\alpha} - Q_\beta$;
9: $\mathcal{D}_\alpha \leftarrow \mathcal{D}_\alpha\backslash\{f_\alpha^\beta\}$, $\mathcal{D} \leftarrow \mathcal{D}\backslash\{f_1^\beta, f_2^\beta \ldots, f_M^\beta\}$.
10: **else**
11: $q_{\alpha\beta} = \frac{U_{temp-\alpha}}{U_\alpha}$;
12: $R_\alpha^\beta = \frac{q_{\alpha\beta}U_\alpha}{Q_\beta}$, $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_\alpha^\beta\}$;
13: $\mathcal{D} \leftarrow \mathcal{D}\backslash\mathcal{D}_\alpha$.
14: **end if**
15: **end while**
16: $Y = \cup_{i\in\mathcal{M}\backslash\{0\}} Y_i$.
17: **for** $f_\alpha^\beta \in \mathcal{W}$, but $f_\alpha^\beta \notin Y$ **do**
18: $R_\alpha^\beta = 0$, $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_\alpha^\beta\}$;
19: **end for**
20: Output R, Y_i, $i \in \mathcal{M}\backslash\{0\}$.



Content	Face	Probability
/sina.com/videos/v1/s1	1	1
/sina.com/videos/v1/s2	2	0.75
...	...	...
/sina.com/videos/v2/s1	4	1

Fig. 5. Communication process of building a routing map $\mathcal{R}$.

the rest element set $\mathcal{D}_\alpha$ will be removed from the ground set. At the end of Algorithm 2, a routing map $\mathcal{R}$ is generated.

Fig. 5 shows the communication process of building a routing map $\mathcal{R}$. As shown in Fig. 5, at the end of each time slot, common routers send their latest content placement decisions (decided by Algorithm 1) to the core router in G_s. Being aware of this information, core router solves Algorithm 2, to calculate what is the uplink bandwidth proportion (*i.e.*, q_{ik}) that each common router should reserve discriminatively for its local cached chunks and to calculate the probability of downstream requests forwarding to a common router (*i.e.*, R_i^k). Based on q_{ik} and R_i^k, core router generates the routing map $\mathcal{R}$ used for next time slot. A sample of routing map $\mathcal{R}$ structure is given in Fig. 5, which is similar with FIB. The map contains three columns: the names of chunks that common routers have cached and reserved uplink bandwidth for; the face of request forwarding; and the probability of implementing such dynamic request forwarding.

Compared with the static routing strategy, which always sends the unsatisfied requests to the original server, dynamic routing strategy brings benefits by using routing map $\mathcal{R}$ to exploit potential downstream content retrieval. For those unsatisfied requests, if there exists a $R_i^k > 0$ in $\mathcal{R}$, they will be sent downstream to the common router N_i according to the probability value R_i^k. Through coordinating all routers in G_s, dynamic request routing strategy satisfies requests in a closer downstream router instead of in the remote server, therefore greatly reduces data access latency and mitigate network traffic.

The cost of dynamic request routing should also be considered, including communication overhead, storage cost and searching cost:

- Communication overhead. It is caused by exchanging content placement information between common routers and core router at each time slot. It depends on the length of time slot and cache size of common routers. For long interval of time slot, as well as small cache size, small communication overhead will be observed.
- Storage cost. It is proportional to the number of items in $\mathcal{R}$. As $\mathcal{R}$ only creates items for the chunks have been cached and reserved uplink bandwidth by common routers in G_s, its storage cost will be much smaller than that of FIB. For each item, its storage cost overhead is 182 bits, (*i.e.*, 166 bits [23] content name + 8 bits forwarding face information + 8 bits probability information).
- Searching cost. It depends on the size of $\mathcal{R}$. Core router will search in $\mathcal{R}$ if it cannot satisfy requests by local CS. As the size of $\mathcal{R}$ is much smaller than that of FIB, it requires less searching cost than FIB. According to [23], 1.3 million lookups per second can be supported by exiting technology. The experiment part will further verify the searching cost of dynamic request routing.

The computational complexity of Algorithm 2 is decided by multiple iterations from step 3 to step 14. Each iteration selects one element f_α^β with highest marginal value from $\mathcal{D}$. In the worst case, if there is no duplicated content placement in all common routers and the bandwidth capacity is abundant, ML iterations are required, where M is the number of common routers in G_s and L is the average number of chunks cached by each common router. In each iteration, the most time consuming is step 3, which requires $O(ML)$ complexity. So the time complexity of Algorithm 2 is less than $O(M^2L^2)$.

5. Performance evaluation

In this section, we evaluate the proposed coordinated content placement and dynamic request routing strategy (termed as CP-Dynamic) and compare it with the following reference schemes:.

- PF-Static (Popularity first + Static request forwarding): each router caches the most popularity videos based on its local statistical information and adopts static request forwarding as the routing strategy.
- MHP-Static (MHP + Static request forwarding): MHP is a short term of Maximum Hit Problem proposed in [2]. According to MHP, the CS of each router is divided into two parts for local and global usages, respectively. The first part caches the most popularity videos based on the local statistics. Then the second part of each router caches the remaining videos collaboratively

Table 2
Summary of simulation parameters.

Meaning	Value
Number of common routers	10
Total number of requests	100,000 (Synthetic), 196,482 (Real)
Total number of video catalog	10,000 (Synthetic), 2,000 (Real)
Video popularity distribution	Zipf, $\alpha \in [0.2, 1.2]$ (Synthetic)
Video size	Uniform distribution, [50MB,200MB] (Synthetic)
Total CSs size	[5%, 45%] of the whole catalog
Cache size ratio	From 1: 5 to 5: 1
Unit cost	$c_0 = 100$, $c_i \in [5, 20]$
Uplink bandwidth capacity	$U_i \in$ [1GB/s,4GB/s]

with only one copy in a subnetwork to ensure maximum coverage of all videos. The optimal proportion of the first and second parts is 6: 4 according to [2]. The static request forwarding is adopted as the routing strategy.

- LRU-Static (Least Recently Used caching + Static request forwarding): In this scheme, each router caches every incoming content and will discard the least recently used items if there is not enough room for the new coming one. The static request forwarding is adopted as the routing strategy.
- LRU-Dynamic (Least Recently Used caching + Dynamic request forwarding): In this scheme, each router caches every incoming content and will replace content according to LRU. The dynamic request forwarding designed in this paper is adopted as the routing strategy.

A discrete event simulation platform is built to implement data caching and request routing and to realize the functions of CS, PIT and FIB modules. The base scenario consists of one core router and 10 common routers. To evaluate the proposed approach and to analyze the impact of content popularity, synthetic traces and real traces are both used:

- Synthetic traces. The video catalog contains 10,000 video files and are requested by 100,000 request events. The video size is uniformly distributed between 50MB and 200MB. We use the Zipf model to formulate the video request pattern with the shape parameter $0.2 \leq \alpha \leq 1.2$, i.e., video k is requested with probability $p_k = \frac{1/k^\alpha}{C}$, $C = \sum_{k=1}^{|K|} 1/k^\alpha$.
- Real traces. We obtain the traces from DBD2 (Delft BitTorrent Data Set 2, which are collected in 5 days by MultiProbe project [24]. First, we use the MaxMind GeoIP database [25] to map them into different areas and select the traces in the area of the United States, which contain 196,482 request events and cover 2000 content files. Then we divide these request events into 11 Autonomous Systems (ASs) by using the regional division method of the United States Census Bureau. Each router in the simulated topology is mapped into one AS and will receive requests of its mapping AS.

In both synthetic and real traces, each video is segmented into chunks of unit size, therefore requests are initialled as chunk-level. That is each video request event will trigger a sequence of chunk-level requests. Each router is equipped with a CS module and the total size of these CSs is ranged from 5% to 45% of the whole catalog. The cache size ratio of core router and common routers is 5: 1. That is the cache size of core router is 5 times the size of common router. Common routers have heterogeneous uplink capacities. The link capacity of U_i is set randomly in the range of [1GB/s, 4GB/s] based on the real-word system. The unit cost c_0 is set to 100 and c_i is in inverse proportional to the corresponding uplink capacity, in the range of [5, 20]. Table 2 summarizes the simulation parameters.

Simulation time is divided into a sequence of time slots. Initially, the Lagrangian multiplier η_k is set to 0 and the routing map $\mathcal{R}$ is empty. At the end of each time slot, η_k is updated and $\mathcal{R}$ is regenerated for dynamic request routing of next time slot.

In our evaluations, five metrics are mainly considered:

- Cost savings: the cost that can be saved by each schemes compared with fetching all videos from the original server.
- Ratio of first cache hit: the average ratio of first cache hit among all routers. The ratio of first cache hit of each router is defined as the ratio of the number of requests it satisfied to the number of requests it received.
- Ratio of supported requests: the ratio of the number of requests satisfied by cache space of all routers to the number of requests launched by end-users.
- Signaling overhead: the ratio of signaling traffic to data traffic in downlink and uplink, respectively. The units of signaling and data traffic volume are B and MB, respectively. For comparison and analysis, the ratio calculation uses their values without regard to units.
- Request forwarding time: the average time necessary to forward each request from the request initial router to the hit router.

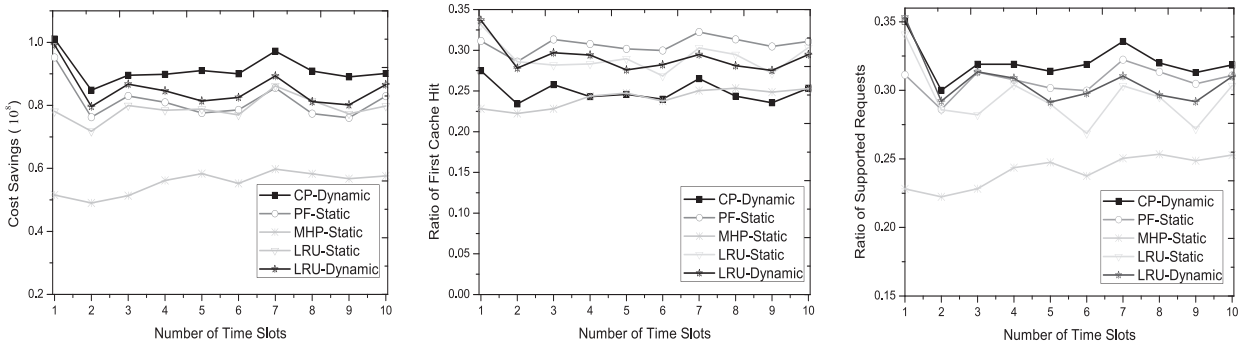
The following simulation results are presented by three parts. First, we evaluate the performance of 4 schemes in basic scenario by using real traces. Second, we will extend our results in discussing the impact of Zipf parameter, cache size capacity and cache size ratio of core router and common routers. Third, we will analyze the parameter settings of uplink bandwidth, number of direct links and Lagrangian multiplier in the proposed model.

5.1. Performance comparison in basic scenario

The basic scenario evaluation uses the real traces. We simple the experimental results at the end of each time slot. The simulation initials 196,842 requests and each time slot includes 19,684 request events. Therefore, there are totally 10 time slots.

Fig. 6(a) shows the average cost savings of each scheme. It can be observed that CP-Dynamic can save 12.4%, 64.9%, 15.8% and 7.4% more costs than that of PF-Static, MHP-Static, LRU-Static and LRU-Dynamic, respectively. The LRU-Dynamic scheme achieves better cost savings than PF-Static scheme and PF-Static performs better than LRU-Static. This also indicates the superiority of dynamic request routing. MHP-Static performs the worst of all, which has two reasons. The first one relates to the cache size allocation between core and common routers and will be discussed in Section 5-D. The second one lies in the request pattern of real traces, where the request arrival of each router is relatively duplicated with each other. In contrast, the majority content in MHP has only one copy cached by all these routers. Therefore, requests in MHP have less probability to be satisfied by the first met router and will be sent to the remote server because of lacking dynamic request routing mechanism, which leads to undesirable cost savings. This can be validated by the relative low ratio of first cache hit and low ratio of supported requests in MHP-Static. The cost savings of CP-Dynamic fluctuates more frequently at the first several rounds of time slot and turns into relative stable at the end of simulation. This is because the value of Lagrangian multiplier η is set randomly as 0 at the beginning and it takes several time slots to be convergent.

Fig. 6(b) shows the average ratio of first cache hit among all routers. A high ratio of first cache hit means that requests are more likely to be satisfied by the first met router. We can see that PF-Static scheme achieves the highest ratio of first cache hit. It is because PF-Static enables each router to cache the local most popular videos according to statistics, therefore can get a high ratio of first cache hit. LRU-Dynamic and LRU-Static have almost the same ratio of first cache hit, as they use the same content caching strategy and will have similar first cache hit events. CP-Dynamic and



(a) Cost Savings. (b) Ratio of First Cache Hit. (c) Ratio of Support Requests.

Fig. 6. Performance comparisons of basic scenario in real traces. The total cache size proportion is 15% of all the content. Number of direct links = 1. The cache size ratio of core and common router is 5:1. $\beta = 1$.

MHP-Static have similar and relative low ratio of first cache hit. As mentioned above, different routers receive similar request sequences in real traces, while both of CP-Dynamic and MHP-Static suggest routers avoid to cache duplicated content. MHP-Static is for maximizing the content cache coverage and CP-Dynamic is for maximizing the cost savings. This leads to relative low ratio of first cache hit. Note that the ratio of first cache hit may be decreased at CP-Dynamic, but the ratio of total cache hit can be guaranteed because of dynamic request routing mechanism. which can be validated by metric evaluation of ratio of supported requests.

Fig. 6(c) shows the ratio of supported requests of each scheme. Higher ratio of supported requests means that more requests can be satisfied by caches of routers rather than sent to the remote original server. So the server load can be greatly reduced. Obviously, CP-Dynamic significantly improves the ratio of supported requests compared with the rest 4 schemes. LRU-Dynamic has only 2% improvement than LRU-Static as the profile of dynamic request routing is not so obvious without the assistance of coordinated content placement. In contrast, through coordination of content placement and dynamic request routing, CP-Dynamic achieves high ratio of supported requests and reduces more server load than the rest 4 schemes. Compared the curves of CP-Dynamic at Fig. 6(b) and (c), we find that CP-Dynamic successfully redirects 10% of requests and satisfies them at nearby routers.

In sum, it can be concluded that (1) CP-Dynamic can save 15.8%, 64.9%, 12.4% and 7.4% more costs than that of PF-Static, MHP-Static, LRU-Static and LRU-Dynamic, respectively. (2) CP-Dynamic may decrease the ratio of first cache hit for maximizing the profit of cost savings, but can firmly guarantee the ratio of total cache hit. Results show that 10% of total requests in CP-Dynamic are successfully redirected to nearby routers rather than sent to the remote server. (3) LRU-Dynamic performs better than LRU-Static, but the improvement is not so obvious as that of CP-Dynamic. This also implies that dynamic request routing should be implemented with coordination of content placement decisions so as to achieve maximum benefit of in-network caching.

5.2. Impact of content popularity

To evaluate the impact of content popularity distribution, Fig. 7 presents the performance comparisons as a function of content popularity. The content popularity in synthetic traces is tuned by the Zipf parameter α. Value of α indicates the popularity concentration degree of video requests. A bigger α means that fewer distinct videos attract majority requests. The α value varies from 0.2 to 1.2 with a step size of 0.2. Therefore, Fig. 7 uses label Zipf (α value) in X-axis to mark results of synthetic traces, which are shaped by different values of α and uses label *Trace* to indicate results from real traces. It can be observed that content popularity significantly affects the performance of each scheme. With α increases, the performance of all schemes becomes better.

Fig. 7(a) shows that with various α values, the proposed CP-Dynamic scheme consistently saves more costs than the rest 4 schemes. As α increases, cost savings gap between CP-Dynamic and PF-Static becomes narrow. It is because that when α is set larger (i.e., $\alpha = 1.2$), the arrival requests of routers are more centralized and the cache capacity of routers in this experiment is set enough large (15% of all the content) to accommodate the most popular videos. Therefore, PU-Static can also achieve a good performance of cost savings by caching the most popular videos.

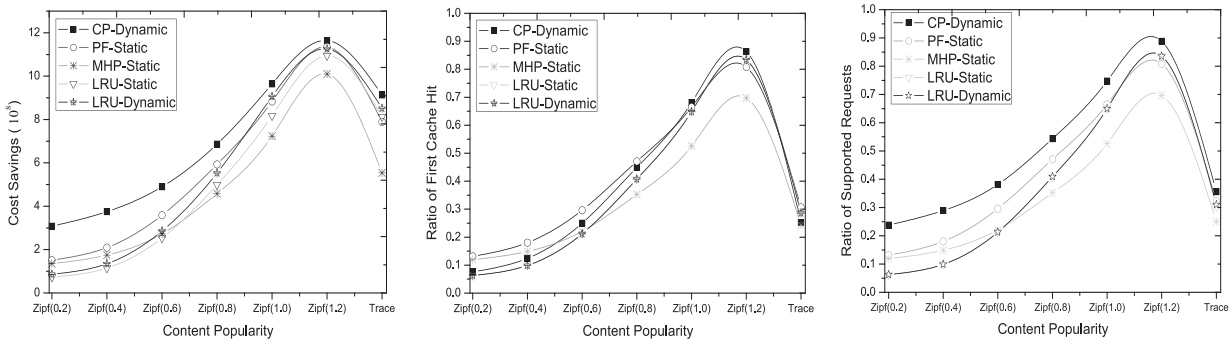
Fig. 7(b) presents that as α increases, the ratio of first cache hit in CP-Dynamic also increases and becomes the best one at $\alpha = 1.0$, $\alpha = 1.2$. Because the content popularity is highly skewed with large α value, the requests of videos become increasingly concentrated. Those requests with the failure of first cache hit have less probability to be satisfied through downstream retrieval. In this case, dynamic routing is less considered and the objective of CP-Dynamic turns into deciding optimal content placement, resulting in high ratio of first cache hit in case of $\alpha = 1.0$, $\alpha = 1.2$.

Fig. 7(c) shows that with various α values, our CP-Dynamic consistently supports more requests than the rest 4 schemes. Specially, when $\alpha = 1.0$, CP-Dynamic can support 12.4%, 42.1%, 15.1% and 13.9% more requests than PF-Static, MHP-Static, LRU-Static and LRU-Dynamic, respectively. Compared the results in synthetic and real traces, it is observed that CP-Dynamic always performs superiorly at cost savings and ratio of supported requests, while may cause performance reduction on ratio of first cache hit in some cases.

In sum, it can be concluded that (1) Content popularity plays a great impact on network performance. The results in synthetic and real traces are similar. As α increases, the performance of all schemes consistently improves. Specially, the performance of LRU-Dynamic becomes more remarkable at highly skewed content popularity (i.e., large α value), while MHP-Static is more applicable for less skewed content popularity. (2) As α increases, the gap of cost savings between CP-Dynamic and the rest schemes becomes narrow. Besides, CP-Dynamic achieves superior ratio of first cache hit at $\alpha = 1.2$. It is because when the requests become more concentrated (i.e., $\alpha = 1.2$), the requirement of dynamic request routing decreases and CP-Dynamic focuses on deciding content placement in this situation. Thus, its ratio of first cache hit improves.

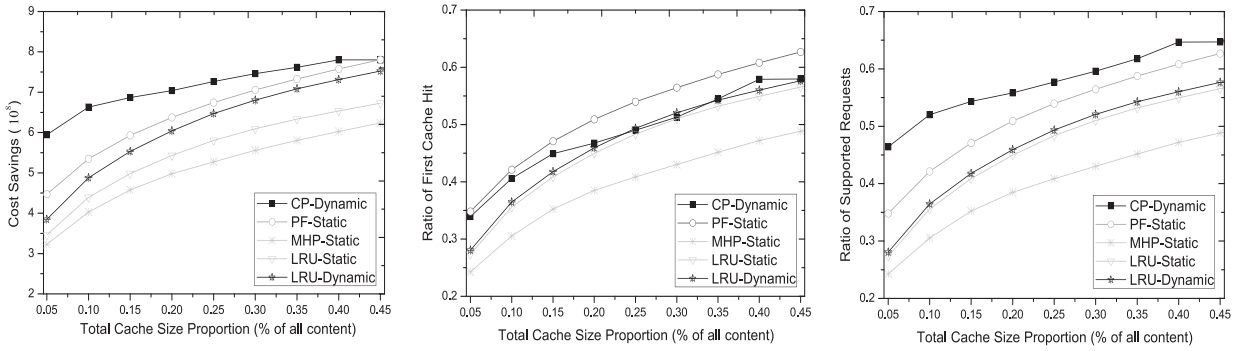
5.3. Impact of total cache size capacity

To evaluate the impact of total cache size capacity, Fig. 8 presents the performance of different schemes. The *x*-axis is the



(a) Cost Savings vs. Content Popularity. (b) Ratio of First Cache Hit vs. Content Popularity. (c) Ratio of Support Requests vs. Content Popularity.

Fig. 7. Performance comparisons as a function of content popularity. The total cache size proportion is 15% of all the content. Number of direct links = 1. The cache size ratio of core and common router is 5:1. $\beta = 1$.



(a) Cost Savings vs. Total Cache Size. (b) Ratio of First Cache Hit vs. Total Cache Size. (c) Ratio of Support Requests vs. Total Cache Size.

Fig. 8. Performance comparisons as a function of total cache size proportion. Zipf parameter $\alpha = 0.8$. Number of direct links = 1. The cache size ratio of core and common router is 5:1. $\beta = 1$.

total cache capacity of all routers, which is a fraction of all content size, ranged from 0.05 to 0.45 of all content with a step size of 0.05.

With total cache capacity increases, the evaluated 3 metrics of each scheme become better. More cache capacity means more videos can be cached by routers and thus can achieve better performance. Another observation is that as the total cache capacity increases, the improvement rate of network performance becomes slower. It indicates that when the total cache capacity increases to certain extent (i.e., 35%), continues increasing cache capacity will less contribute to network performance.

As shown in Fig. 8(a), CP-Dynamic can always save more costs than the rest schemes but the gap between CP-Dynamic and PF-Static becomes narrow at a large cache capacity setting. It is because that when the total cache capacity is large enough (e.g., 45% of all videos), PF-Static can achieve a considerate good performance near to that of CP-Dynamic by caching the most popular videos. However, for the most realistic cache capacity settings (from 5% to 30%), CP-Dynamic significantly outperforms PF-Static. When total cache capacity is 15% of all videos, CP-Dynamic can save 16.9%, 49.9%, 37.4% and 24.1% more costs than PF-Static, MHP-Static, LRU-Static and LRU-Dynamic schemes, respectively.

Fig. 8(b) shows the ratio of first cache hit and Fig. 8(c) presents the ratio of supported requests of these schemes. It can be seen that CP-Dynamic has smaller ratio of first cache hit with that of PF-Static but can support more requests than PF-Static. It is because that CP-Dynamic aims to coordinate all the routers in a subnetwork to achieve favorable cost savings, therefore encourages to less duplicated video storage. In CP-Dynamic, requests may not be satisfied by the first met routers but will be hit by a nearby router. In this coordination manner, more requests can be satisfied by the in-network caching capacity of routers, rather than be sent to the remote original server.

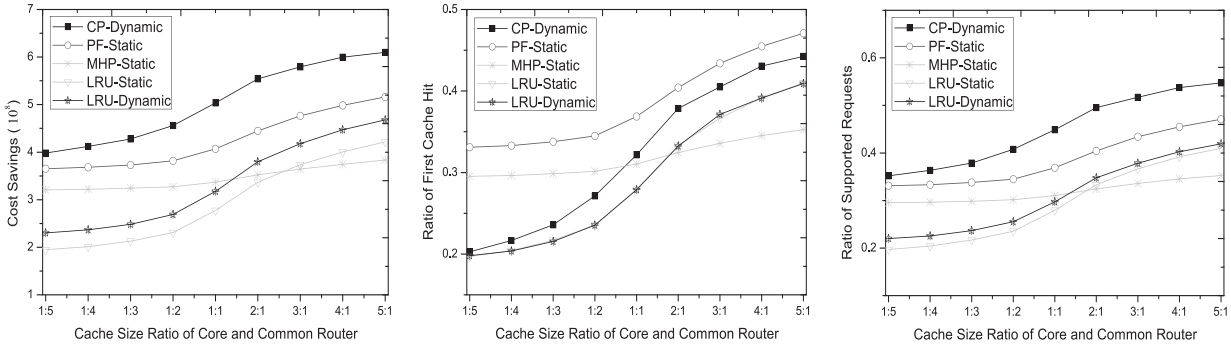
In sum, it can be concluded that (1) the performance of each scheme will be affected by total cache size capacity, but the impact decreases when the cache capacity increases to certain extent. (2) The benefit of CP-Dynamic is more remarkable at limited total cache size capacity.

5.4. Impact of cache size ratio of core and common routers

Cache size ratio of core and common routers is defined as the cache capacity allocation ratio between the core router and single common router given total cache size capacity. Next, we call it as core/common ratio for short. Fig. 9 presents the performance comparisons of different schemes when the core/common ratio ranges from 1: 5 to 5: 1.

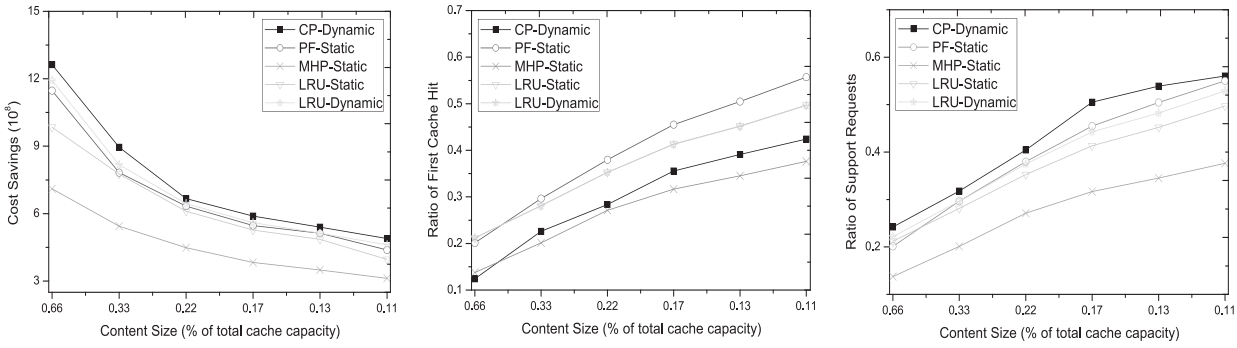
One observation is that cache capacity allocation between core and common routers has a direct impact on the network performance. Performance of all the 5 schemes becomes better when core/common ratio changes from 1: 5 to 5: 1. Fig. 9(a) shows the cost savings on various core/common ratios and it can be seen that the cost savings advantage of CP-Dynamic becomes more remarkable at 5: 1 core/common ratio setting.

When core/common ratio changes, the performance fluctuation of CP-Dynamic is obvious, especially at the ratio of first cache hit. When the core/common ratio is small (e.g., 1: 5), it means that common routers are assigned with more cache size. Therefore, CP-Dynamic has a higher probability to redirect requests to another common routers in a subnetwork. Consequently, the ratio of first



(a) Cost Savings vs. Core:Common. (b) Ratio of First Cache Hit vs. Core:Common. (c) Ratio of Support Requests vs. Core:Common.

Fig. 9. Performance comparisons as a function of cache size ratio of core and common routers. The total cache size proportion is 15% of all the content. Zipf Parameter $\alpha = 0.8$. Number of direct links $= 1$. $\beta = 1$.



(a) Cost Savings vs. Content Size. (b) Ratio of First Cache Hit vs. Content Size. (c) Ratio of Support Requests vs. Content Size.

Fig. 10. Performance comparisons as a function of content size in real traces. The cache size ratio of core and common router is 5:1. Number of direct links $= 1$. $\beta = 1$.

cache hit is low. While when the core/common ratio becomes large (e.g., 5: 1), the cache size of core router is much larger than that of common router. In this situation, core router can satisfy more requests and those unsatisfied requests at core routers have less probability to be redirected. Thus, the ratio of first cache hit greatly improves.

Another observation is that MHP-Static has better performance than LRU-Static and LRU-Dynamic at small core/common ratio, but this observation reverses when the core/common ratio is larger than 3: 1. In experimental scenarios, core router receives external requests from users and those unsatisfied requests at common routers. Therefore, allocating more cache size to core router together with the assistance of LRU policy can simply achieve a relative good network performance compared with MHP-Static.

In sum, it is concluded that (1) Compared with total cache size capacity, the cache size allocation between core and common routers plays a more effective impact on the schemes. (2) The performance of all the 5 schemes becomes better when core/common ratio changes from 1: 5 to 5: 1. (3) CP-Dynamic shows distinguished advantages when core router is allocated more cache size.

5.5. Impact of content size

Fig. 10 presents the impact of content size on different schemes. To evaluate the impact of content size, we keep the total cache capacity constant and vary the average size of each content from 0.66% to 0.11% of total cache capacity.

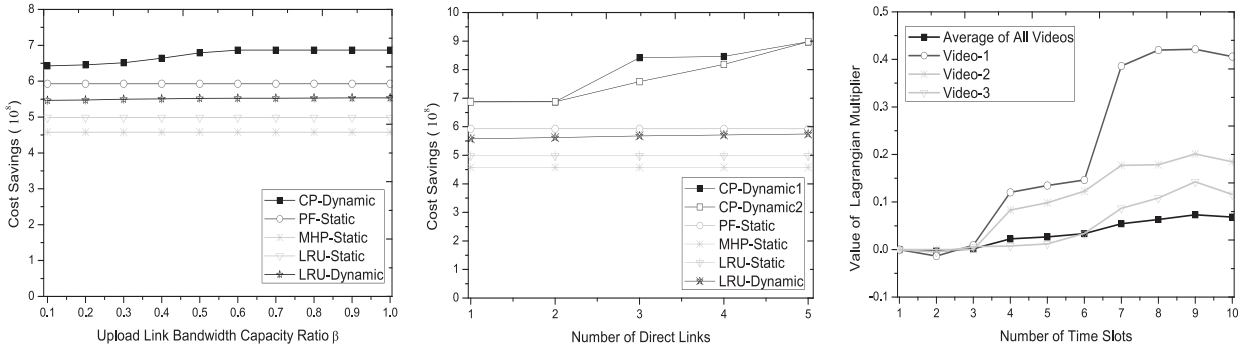
Obviously, the smaller content size is, the more content items can be cached by routers. For accommodating more content items, routers have higher probability to satisfy requests by the first cache hit and can support more requests rather than sending them to the original content server. Therefore, as the content size decreases, the ratio of first cache hit (see Fig. 10(b)) and the ratio of support requests (see Fig. 10(c)) of all schemes are all improved.

Fig. 10(a) shows that as content size decreases, the cost savings of all schemes reduces. The cost savings is proportional to the content size, and thus the decline of content size will lead to the reduction of cost savings. Notice that the curves of cost savings are less skewed with small cache size. As explained above, as the content size becomes smaller, the ratio of support requests improves, which is beneficial for cost savings and reduces the loss of cost savings brought by decreased content size.

5.6. Impact of related parameters settings

In the following, we analyze the parameter settings of uplink bandwidth capacity, number of direct links and Lagrangian multiplier η.

To evaluate the impact of uplink bandwidth capacity, the uplink bandwidth of each common router is set to βU_i, where U_i is set randomly in the range of [1GB/s, 4GB/s] and β is a tuned factor. Fig. 11(a) shows the cost savings of different schemes when U_i has already been determined for each router and β ranges from 0.1 to 1.0 with a step size 0.1. It can be observed that as β increases from 0.1 to 0.6, the cost savings of CP-Dynamic also increase steadily. This indicates that incremental uplink bandwidth capacity is used to support more dynamic request routing and thus can lead to more cost savings. When β increases from 0.7 to 1.0, the cost savings of CP-Dynamic stay unchanged. It is because that when β is set to 0.7, the uplink bandwidth capacity is large enough to support all the dynamic request routing requirements in experiments. So continuous increasing β would not cause more cost savings. With β increases, the cost savings of LRU-Dynamic have a slight improvement. The reason is that without coordination of



(a) Cost Savings vs. β Parameter. (b) Cost Savings vs. Number of Direct Links. (c) Cost Savings vs. η Parameter.

Fig. 11. Impact of parameter settings. Zipf parameter $\alpha = 0.8$. The total cache size proportion is 15% of all the content. The cache size ratio of core and common router is 5:1.

content placement, the requirement of dynamic request routing is limited and 10% of U_i is almost enough for LRU-Dynamic. The caves of PF-Static, MHP-Static and LRU-Static stay unchanged because no uplink bandwidth capacity is required in these schemes.

To evaluate the impact of number of direct links, we randomly connect two common routers and increase the number of router connections. Fig. 11(b) shows the cost savings of different schemes when the number of direct links increases from 1 to 5. To avoid the impact of random router connections, the experiment runs CP-Dynamic under two types of direct link connection topologies, termed as CP-Dynamic-1 and CP-Dynamic-2. It can be observed that as the number of direct links increases, the cost savings of CP-Dynamic-1 and CP-Dynamic-2 significantly improve. According to Theorem 1, interconnected common routers can coordinate as one single router with aggregated storage capacities. Therefore, increasing number of direct links improves the coordination among routers and thus causes more cost savings. The caves of PF-Static, MHP-Static and LRU-Static stay unchanged because they do not consider coordination between interconnected routers.

Finally, we examine the relation between Lagrangian multiplier and the content popularity. η_{ik} represents the Lagrangian multiplier of video k at router N_i. In the experiments, we sample each η_{ik} at the end of each time slot. Fig. 11(c) shows the average Lagrangian multiplier value of all η_{ik} (termed as Average of All Videos) and the Lagrangian multiplier values of the most unpopular videos at core router. Video-1 is the most unpopular video at core router, video-2 is the second and video-3 is the third unpopular one. It can be seen that all the Lagrangian multiplier values are initialed as 0, fluctuate with time and turns into stable at the end of 10-*th* time slot. This indicates that Lagrangian multiplier takes time to be convergence and will adjust according the volatility of data. Another observation is that the less unpopular the video is, the larger its Lagrangian multiplier value will be. The value of video-1 is larger than video-2 and the value of video-3 is the smallest one. This conforms to the meaning of η_{ik}, which reflects the capacity of video sharing and will be increased for insufficient storage and bandwidth provision for the corresponding video.

5.7. Request forwarding time

Here, we examine the line rate operation effectiveness of different schemes on an Intel(R) Core(TM) i7-2600 CPU @ 3.4 GHz machine with 4GB of RAM and use request forwarding time as the measurement metric. Upon receiving a request, a router need to check its modules (i.e., CS, PIT and FIB) to decide whether this request can be satisfied locally and where to forward this request. Less request forwarding time means higher effectiveness of line rate operation. For schemes of PF-Static, MHP-Static and LRU-Static, the request forwarding time is the delay caused by searching in CS, PIT and FIB. While CP-Dynamic and LRU-Dynamic may require extra searching time in their built dynamic routing map $\mathcal{R}$.

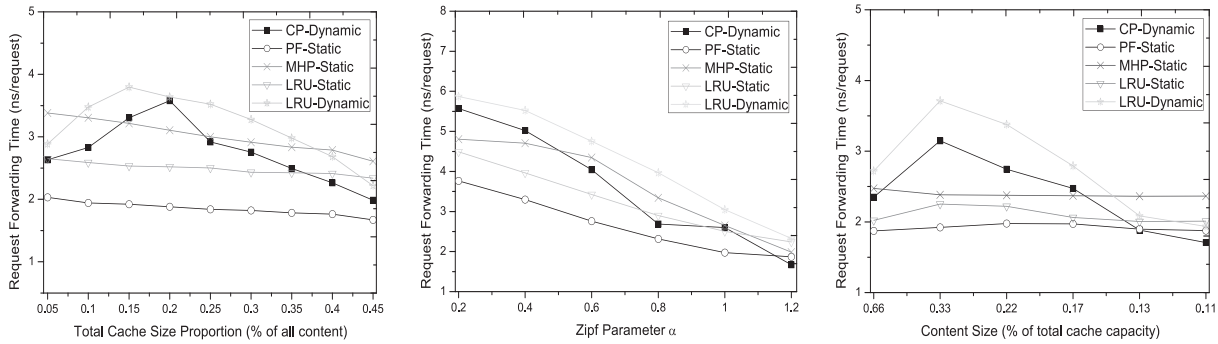
As the request forwarding time of each request is not fixed and it may fluctuate due to various cache status of CS and routing path, the experiments test the time required to forward all the requests and then calculate the average request forwarding time per request (in unit of *ns/request*).

Fig. 12(a) plots the request forwarding time of different schemes as a function of total cache size. Obviously, PF-Static requires the minimum request forwarding time. It is because PF-Static achieves higher ratio of first cache hit than MHP-Static and LRU-Static, and requires no an extra search time in dynamic routing map as LRU-Dynamic and CP-Dynamic. Generally, with total cache capacity increased, the search time in CS will also increase as larger size of cache space needs to be searched. However, the request forwarding time of PF-Static, MHP-Static and LRU-Static decreases. The reason lies in that the increased cache capacity may cause more retrieval time in CS, but may also enable it to satisfy requests within fewer number of request forwarding. Taken together, these two factors lead to a slight decline of request forwarding time in PF-Static, MHP-Static and LRU-Static.

For LRU-Dynamic and CP-Dynamic, their request forwarding time first increases when total cache capacity ranges from 5% to 20%, while turns into decline afterwards. In the range from 5% to 20% of total cache capacity, the increased searching time in CS and $\mathcal{R}$ is the main factor for increasing request forwarding time. However, when the total cache capacity increases to large enough, the searching in CS still increases, but the retrieval in $\mathcal{R}$ might be reduced as requests might have been satisfied by CS, or the retrieval in FIB might be reduced if requests have been hit by $\mathcal{R}$. The extra search time in $\mathcal{R}$ is much less than the search time in FIB. Therefore, the request forwarding time begins to decrease.

Fig. 12(b) plots the request forwarding time of different schemes as a function of Zipf parameter α. It is observed that the request forwarding time of all schemes decreases as α value becomes larger. This behavior is caused by the fact that for large α value, the content popularity is more centralized and thus routers have higher probability to satisfy requests without forwarding them to the next hop. This leads to the decline of request forwarding time.

Fig. 12(c) plots the request forwarding time of different schemes as a function of content size. Obviously, when content size becomes smaller, more content items can be cached by routers, leading to more retrieval time in CS and $\mathcal{R}$. This is the reason why the request forwarding time of LRU-Dynamic and CP-Dynamic increases in the range from 0.66% to 0.33%. When the content size



(a) Request Forwarding Time vs. Total Cache Size. (b) Request Forwarding Time vs. Zipf Parameter α. (c) Request Forwarding Time vs. Content Size.

Fig. 12. Request forwarding time plots. Number of direct links = 1. The cache size ratio of core and common router is 5:1. $\beta = 1$.

reduces continuously (from 0.33% to 0.11%), routers accommodate enough number of content to satisfy requests. In this situation, the retrieval time of CS still increases, but the frequency of searching in $\mathcal{R}$ will reduce, which causes the request forwarding time to decrease. Furthermore, the curves of PF-Static, MHP-Static and LRU-Static fluctuate slightly with various content size. As explained above, the decreased content size enables routers to cache more content. This causes more retrieval time in CS, but also reduces the number of request forwarding. These two factors lead to the curves fluctuation of PF-Static, MHP-Static and LRU-Static.

In sum, the cause of request forwarding time's fluctuation is complicated and is a consequence of many factors, such as total cache capacity, content popularity and content size. In specific, it mainly depends on whether the requests can be hit by the modules of CS, PIT, $\mathcal{R}$ and FIB. With respect to CP-Dynamic, it requires extra search time in $\mathcal{R}$, but may reduce a further search operation in FIB if requests have been hit by $\mathcal{R}$. As discussed at the end of Section 3, the size of $\mathcal{R}$ is much smaller than that of FIB. CP-Dynamic pays for less search time in $\mathcal{R}$ but reduces the much more search time in FIB. Therefore, it is worthy to take extra search operation in $\mathcal{R}$, which will not increase burden on the line rate request forwarding. For example, with large α value, as well as small content size, CP-Dynamic consumes the minimum request forwarding time compared with the rest of 4 schemes.

5.8. Signaling overhead

Finally, we evaluate the signaling overhead that CP-Dynamic brings to the network.

At the end of each time slot, signaling messages are exchanged between common routers and core router in a subnetwork for the purpose of updating cache status in these routers and generating a routing map $\mathcal{R}$ in core router. Common routers send uplink messages to inform core router about the necessary information of their received requests, while core router sends downlink messages to tell common routers how to update their cache status after running Algorithms 1 and 2.

Fig. 13 shows the uplink and downlink signaling overhead of CP-Dynamic as a function of total request arrival rate. The total request arrival rate varies from 10 requests per second to 100 requests per second. The impact of time slot length on signaling overhead is also evaluated. We set the time slot length from 0.5 to 2 min, termed as 0.5 min, 1.0 min, 1.5 min and 2.0 min, respectively.

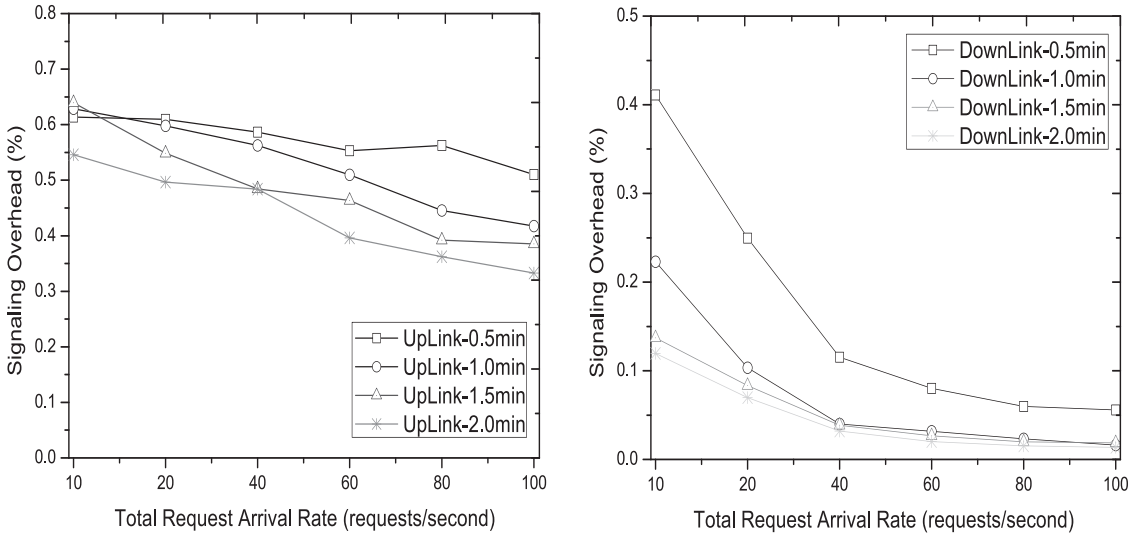
For fast request arrival rate, as well as long time slot interval, more data is transferred in the network and therefore results in less downlink signaling overhead (see Fig. 13(b)), since the number of downlink messages is independent of request arrival rate and time slot length, but mainly related to the number of common routers. Furthermore, in case of fast request arrival rate and long time slot interval, besides of more data is transferred, more information about requests will be collected by common routers and then causes more uplink messages. Because the increase of uplink messages is less than the increased amount of data, uplink signaling overhead still decreases in this case (see Fig. 13(a)), but its curve is less skewed than that of downlink signaling overhead.

In sum, CP-Dynamic brings signaling overhead to the network for the necessary of information collection and realizing coordination between routers. It is also observed that both of uplink and downlink signaling overhead are not heavy as signaling messages only exchange between core router and common routers in a same subnetwork at the end of each time slot. No signaling messages are required during the time slot interval. In general, uplink signaling overhead is slightly more than downlink signaling overhead.

6. Summary of findings

The previous section has explored the effectiveness of coordinated content placement and request routing, and the key factors that impact the coordination of in-network caching. Here, we drew a number of generation conclusions from our study:

- On average, CP-Dynamic can save 12.4%, 64.9%, 15.8% and 7.4% more costs than that of PF-Static, MHP-Static, LRU-Static and LRU-Dynamic, respectively. This shows the effectiveness of coordinating content placement and request routing in CCN.
- Content popularity has a significant impact on the content placement and request routing strategy. As the requests become more concentrated, the performance of each caching scheme improves.
- The factor of total cache size capacity will affect the performance of each scheme, but its impact decreases when the total cache capacity increases to some extent. Compared with the impact of total cache size capacity, the impact of cache size allocation between core and common routers is more remarkable. This indicates that it should dedicated more attentions to cache size allocations between routers.
- Dynamic request routing can improve network performance by finding a copy of the requested content at nearby routers. But its benefit is not so obvious when it is decoupled from content placement decisions (See comparison between LRU-Dynamic and LRU-Static). Without the coordination of content placement, neighboring routers may have similar content cache status. A request that is unsatisfied at one router has less probability to be hit by a nearby router. Therefore, the potential of dynamic request routing is restricted.



(a) Uplink Signaling Overhead. (b) Downlink Signaling Overhead.

Fig. 13. Signaling overhead plots. The total cache size proportion is 15% of all the content. Number of direct links = 1. The cache size ratio of core and common router is 5:1. $\beta = 1$.

- CP-Dynamic achieves more remarkable network performance under limited total cache size capacity scenario and prefers more cache size to be allocated to core routers. When the total cache size capacity is large enough, the simple LRU-Static can achieve a considerate good network performance.

7. Conclusion

In this paper, we investigate the characteristics of *arbitrary network topology* and *dynamic request routing requirement* of CCN in-network caching. In response to these characteristics, we propose a complete model to formulate the coupled content placement and request routing problem under link bandwidth and cache space constraints, aiming at maximizing the bandwidth cost savings. Experiments validate the effectiveness of the proposed model and show that content popularity, cache size allocation among routers, total cache size capacity and number of direct links are important factors and significantly affects the network performance.

There are several directions for future works. First, as we mentioned earlier, the in-network caching system is highly dynamic. It is thus wise to investigate how the temporal dynamics impact the coordination of content placement and dynamic request routing. Second, our work replies on the length of each time slot and the setting of Lagrangian multiplier. Therefore, it is important yet challenging to analyze the impact of time slot division on the convergence of Lagrangian multiplier.

Appendix A

Proof of Theorem 3. Denote the ground set of subproblem (I) as $\mathcal{S} = \{f_0^1, f_0^2, \ldots, f_0^K, f_1^1, f_1^2, \ldots, f_M^K\}$. Obviously, any subset of $\mathcal{S}$ represent a caching state. Suppose that A and B are caching states and satisfy $A, B \subseteq \mathcal{S}$, $A \subseteq B$. Therefore, for any $f_i^k \in \mathcal{S} \backslash B$, we need to prove that:

$$g(A \cup \{f_i^k\}) - g(A) \geq g(B \cup \{f_i^k\}) - g(B), i \in \mathcal{M} \tag{24}$$

In order to prove Eq. (24), we consider two cases based on different caching locations of chunk k.

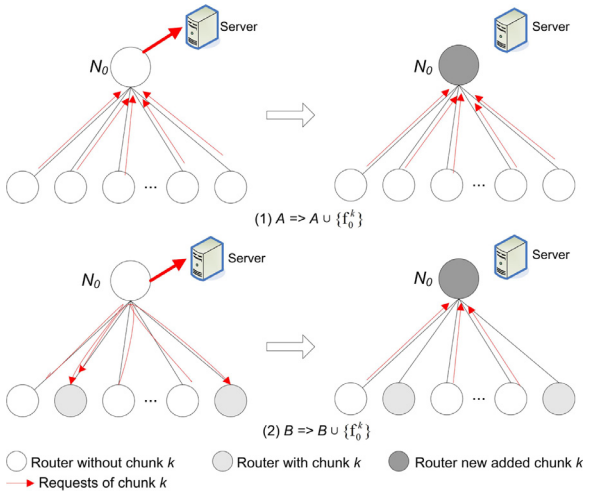


Fig. 14. Illustration of marginal value calculation of $A \cup \{f_0^k\}$ and $B \cup \{f_0^k\}$.

A1. Case 1: consider the placement of chunk k *at core router.*

Through calculating the marginal value brought by adding chunk k to core router N_0, we prove $g(A \cup \{f_0^k\}) - g(A) \geq g(B \cup \{f_0^k\}) - g(B)$ and 3 situations are discussed:

1. *Chunk k is cached by none of the routers in A and B.* In this case, requests of chunk k in G_s will all be directed to original server. In consideration of adding f_0^k to A and B, it means to cache chunk k at core router N_0. Then the marginal value (i.e., the cost saving) brought by f_0^k is the same for A and B because they both save the cost from N_0 to original server. Thus, we have $g(A \cup \{f_0^k\}) - g(A) = g(B \cup \{f_0^k\}) - g(B) = \sum_{i \in \mathcal{M}} \lambda_{ik} sc_0$.
2. *Chunk k is cached at B but not at A.* By adding f_0^k to A, all requests of chunk k in G_s that previously would be directed to original server can be satisfied at N_0. Therefore, the marginal value brought by adding f_0^k to A is $g(A \cup \{f_0^k\}) - g(A) = \sum_{i \in \mathcal{M}} \lambda_{ik} sc_0$. Fig. 14(1) shows the marginal value calculation in this case. As for adding f_0^k to B, if f_0^k already ex-

ists at B, then $g(B\cup\{f_0^k\})-g(B)=0$. Otherwise, chunk k is not cached by N_0 but by some common router N_j at B and Fig. 14(2) shows the marginal value calculation in this case.. Let J represents the set of common routers that cache chunk k, $\forall j\in J$, $f_j^k\in B$. Then the marginal value brought by adding f_0^k to B is $\sum_{i\in\mathcal{M}\setminus J}\lambda_{ik}s(c_{ij}-c_i)$, which means that the unsatisfied requests at $N_i, i\in\mathcal{M}\setminus J$ that previously had to be sent to a nearest common router at J will be satisfied by core router in shorter distance. c_{ij} denotes the unit cost of sending chunks from N_i to a nearest common router at J and $(c_{ij}-c_i)<c_0$ exists as shown in Section 3. Therefore, the marginal value brought by adding f_0^k to A (i.e., $\sum_{i\in\mathcal{M}}\lambda_{ik}sc_0$) is larger than B (i.e., $\sum_{i\in\mathcal{M}\setminus J}\lambda_{ik}s(c_{ij}-c_i), J=\{N_j: f_j^k\in B\}$). That is $g(A\cup\{f_0^k\})-g(A)>g(B\cup\{f_0^k\})-g(B)$,

3. *Chunk k has already been cached by A and B.(i)* If f_0^k already exists at A and B, then the marginal value brought by adding f_0^k to A and B are both equal to 0. *(ii)* If f_0^k exists neither at A nor B, and chunk k is cached by some common router N_j, $j\in J$. Since $A\subseteq B$, obviously cache router that cache chunk k at A also cache chunk k at B. If A and B have the same number copies of chunk k, the number of requests of chunk k received by core router at A and B will be the same, which is $\sum_{i\in\mathcal{M}\setminus J}\lambda_{ik}$. Then the marginal value of adding f_0^k to A and B is also the same, i.e., $g(A\cup\{f_0^k\})-g(A)=g(B\cup\{f_0^k\})-g(B)=\sum_{i\in\mathcal{M}\setminus J}\lambda_{ik}s(c_{ij}-c_i)$. Otherwise, B has more number copies of chunk k than A. Suppose that A and B have the same request patterns, the number of requests of chunk k received by core router at A is more than B, then the marginal value of adding f_0^k to A is larger than that of B, i.e., $g(A\cup\{f_0^k\})-g(A)>g(B\cup\{f_0^k\})-g(B)$. *(iii)* If f_0^k exists at B but not A, $g(B\cup\{f_0^k\})-g(B)=0$ and $g(A\cup\{f_0^k\})-g(A)=\sum_{i\in\mathcal{M}\setminus J}\lambda_{ik}s(c_{ij}-c_i)$, similar to the discussion above. To sum up *(i)*, *(ii)* and *(iii)*, we have $g(A\cup\{f_0^k\})-g(A)\geq g(B\cup\{f_0^k\})-g(B)$.

According to the above 3 situations analysis, $g(A\cup\{f_0^k\})-g(A)\geq g(B\cup\{f_0^k\})-g(B)$ consistently holds in case 1.

A2. Case 2: consider the placement of chunk k *at common router.*

Here we need to prove $g(A\cup\{f_i^k\})-g(A)\geq g(B\cup\{f_i^k\})-g(B), i\in\mathcal{M}\setminus\{0\}$. There are also 3 situations to be discussed.

1. *Chunk k is cached by the core router both at A and B.* In this case, adding f_i^k to A or B only affects the requests at common router N_i. Therefore, $g(A\cup\{f_i^k\})-g(A)=g(B\cup\{f_i^k\})-g(B)=\lambda_{ik}sc_i$.
2. *Chunk k is not cached by the core router neither at A nor at B.* Since $A\subseteq B$, the common routers that cache chunk k at A also cache chunk k at B. *(i)* If A and B have the same number copies of chunk k, the request routing strategies of chunk k at A and B will be the same. Therefore, the number of requests of chunk k directed to common router N_i will be also the same at A and B after adding chunk k at N_i. So we have $g(A\cup\{f_i^k\})-g(A)=g(B\cup\{f_i^k\})-g(B)$. *(ii)* If B has more number copies of chunk k than that of A, obviously, the number of requests directed to common router N_i at B will be less than that of A. Fig. 15 shows an illustration in this case. Thus, $g(A\cup\{f_i^k\})-g(A)>g(B\cup\{f_i^k\})-g(B)$. To sum up (1) and (2), $g(A\cup\{f_i^k\})-g(A)\geq g(B\cup\{f_i^k\})-g(B)$ still holds.
3. *Chunk k is cached by the core router at B but not at A.* In this case, adding f_i^k to B only affects the requests at common router N_i, which will be satisfied locally rather than sent to the core router, i.e., $g(B\cup\{f_i^k\})-g(B)=\lambda_{ik}sc_i$. As for A, because its core router does not hold chunk k, adding f_i^k will not only satisfy requests at common router N_i but also satisfy requests from other routers, in which routers it is the best choice to forward requests of chunk k to N_i. Therefore, $g(A\cup\{f_i^k\})-g(A)>g(B\cup\{f_i^k\})-g(B)$.

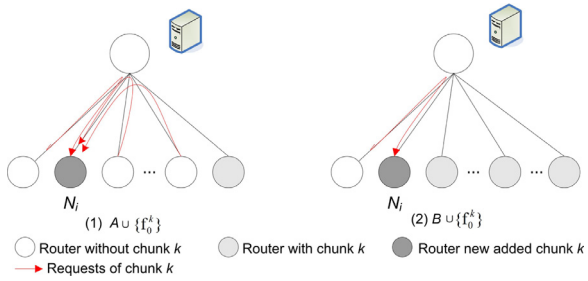


Fig. 15. Illustration of marginal value calculation of $A\cup\{f_i^k\}$ and $B\cup\{f_i^k\}$.

In case 2, $g(A\cup\{f_i^k\})-g(A)\geq g(B\cup\{f_i^k\})-g(B),\quad i\in\mathcal{M}\setminus\{0\}$ consistently holds.

In sum of case 1 and case 2,

$$g(A\cup\{f_i^k\})-g(A)\geq g(B\cup\{f_i^k\})-g(B), i\in\mathcal{M}$$

always holds. Therefore, the objective of subproblem (I) is submodular.

References

[1] V. Jacobson, D. Smetters, J. Thornton, M.F. Plass, N. Briggs, R. Braynard, Networking named content, ACM CoNEXT, ACM, 2009.
[2] J. He, H. Zhang, B. Zhao, S. Rangarajan, A collaborative framework for in-network video caching in mobile networks, 2014, ArXiv preprint arXiv:1404.1108.
[3] P. Radoslavov, R. Govindan, D. Estrin, Topology-informed internet replica placement, Comput. Commun. 25 (4) (2002) 384–392.
[4] S. Borst, V. Gupta, A. Walid, Distributed caching algorithms for content distribution networks, IEEE INFOCOM, 2010.
[5] Y. Xu, Y. Li, T. Lin, G. Zhang, Z. Wang, S. Ci, A dominating-set-based collaborative caching with request routing in content centric networking, IEEE ICC, 2013.
[6] R. Chiocchetti, D. Rossi, G. Rossini, G. Carofiglio, D. Perino, Exploit the known or explore the unknown?: hamlet-like doubts in icn, in: Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking, ACM, 2012, pp. 7–12.
[7] G. Carofiglio, M. Gallo, L. Muscariello, Bandwidth and Storage Sharing Performance in Information Centric Networking.
[8] R. Elisha, On the analysis and management of cache networks, in: Dissertations Paper, 2012, pp. 1–100.
[9] V.G.S. Borst, A. Walid, Distributed caching algorithms for content distribution networks, in: INFOCOM 2010, IEEE, 2010, pp. 302–311.
[10] J. Dai, Z. Hu, B. Li, J. Liu, B. Li, Collaborative hierarchical caching with dynamic request routing for massive content distribution, in: IEEE INFOCOM, 2012, pp. 2444–2452.
[11] W. Jiang, S. Ioannidis, L. Massoulié, F. Picconi, Orchestrating massively distributed cdns, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, ACM, 2012, pp. 133–144.
[12] L. Saino, I. Psaras, G. Pavlou, Hash-routing schemes for information centric networking, in: Proceedings of the 3rd ACM SIGCOMM Workshop on Information–Centric Networking, ACM, 2013, pp. 27–32.
[13] T. Kawano, M. Shimamura, H. Koga, A selective caching scheme based on request history in content-centric networks, in: Proceedings of the 2013 Workshop on Student Workhop, ACM, 2013, pp. 47–48.
[14] S. Wang, J. Bi, J. Wu, Collaborative caching based on hash-routing for information-centric networking, in: ACM SIGCOMM Computer Communication Review, 43, ACM, 2013, pp. 535–536. no. 4
[15] Y. Li, T. Lin, H. Tang, P. Sun, A chunk caching location and searching scheme in content centric networking, in: Communications (ICC), 2012 IEEE International Conference on, IEEE, 2012, pp. 2655–2659.
[16] S. Eum, K. Nakauchi, M. Murata, Y. Shoji, N. Nishinaga, Potential based routing as a secondary best-effort routing for information centric networking (icn), Comput. Netw. 57 (no. 16) (2013) 3154–3164.
[17] R. Chiocchetti, D. Perino, G. Carofiglio, D. Rossi, G. Rossini, Inform: a dynamic interest forwarding mechanism for information centric networking, in: Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking, ACM, 2013, pp. 9–14.

[18] Y. Wang, K. Lee, B. Venkataraman, R.L. Shamanna, I. Rhee, S. Yang, Advertising cached contents in the control plane: necessity and feasibility, in: Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on, IEEE, 2012, pp. 286–291.
[19] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, L. Zhang, A case for stateful forwarding plane, Comput. Commun. 36 (no. 7) (2013) 779–791.
[20] Y. Xu, Y. Li, T. Lin, G. Zhang, Z. Wang, S. Ci, A dominating-set-based collaborative caching with request routing in content centric networking, in: ICC2013, IEEE, 2013, pp. 3624–3628.
[21] M. Akbar, E. Manning, G. Shoja, S. Khan, Heuristic solutions for the multiple-choice multi-dimension knapsack problem, in: Computational Science-ICCS 2001, Springer, 2001, pp. 659–668.
[22] G.L. Nemhauser, L.A. Wolsey, M.L. Fisher, An analysis of approximations for maximizing submodular set functionsłi, Math. Program. 14 (no. 1) (1978) 265–294.
[23] Y. Wang, K. He, H. Dai, W. Meng, J. Jiang, B. Liu, Y. Chen, Scalable name lookup in ndn using effective name component encoding, in: 2012 ICDCS, IEEE, 2012, pp. 688–697.
[24] A. Iosup, P. Garbacki, J. Pouwelse, D. Epema, Multiprobe Project, 2013.
[25] MaxMind, Maxmind Geoip Database, 2013.

Yuemei Xu is a lecturer in the department of Computer Sciences, Beijing Foreign Studies University. She received her the Ph.D. degree at Chinese Academy of Sciences in 2014, the B.E. at Beijing University of Posts and Telecommunications (China) in 2009. Her main research interests include caching optimization in Content Centric Networking and future Internet.

Song Ci is an associate professor in the Department of Electrical and Computer Engineering at the University of Nebraska-Lincoln, USA. He is the Director of the Intelligent Ubiquitous Computing Lab (iUbiComp Lab) and holds a courtesy appointment of UNL Ph.D. in the Biomedical Engineering Program. His research interests include dynamic complex system modeling and optimization, green computing and power management.

Yang Li received the Ph.D. degree in Communication Networking in Kyungpook National University, Korean, in 2009. She is currently an associate professor in Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include content distribution, future Internet, and security in ICN.

Tao Lin is an associate professor of the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences (CAS). His research interests include information/content centric network, in-network caching, mobile computing. He has published more than30 papers in those areas.

Gang Li is a senior lecturer in School of Information Technology, Deakin University, Australia. His research interests include the area of artificial intelligence, more particularly in the area of machine learning and data mining. He is working with a couple of High Degree by Research students, in the TULIP (Team for Universal Learning and Intelligent Processing) Lab.