Computer Networks 110 (2016) 1–17

Contents lists available at ScienceDirect

Computer Networks

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

A minimum cost cache management framework for information-centric networks with network coding[☆]

Jin Wang[a], Jing Ren[b], Kejie Lu[c,d,*], Jianping Wang[e], Shucheng Liu[f], Cedric Westphal[f,g]

[a] *Department of Computer Science and Technology, Soochow University, No.1 Shizi Road, Suzhou, 215006, China*
[b] *School of Communication and Information Engineering, University of Electronic Science and Technology of China, 2006 Xiyuan Avenue, Chengdu, 611731, China*
[c] *College of Computer Science and Technology, Shanghai University of Electronic Power, No.2588 Changyang Road, Yangpu District, Shanghai, 200090, China*
[d] *Department of Electrical and Computer Engineering, University of Puerto Rico at Mayagüez, Mayagüez, Puerto Rico 00681, USA*
[e] *Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong*
[f] *Huawei Technologies Co.,Ltd, Bantian, Longgang DIST, Shenzhen, 518129, China*
[g] *Department of Computer Engineering, University of California, Santa Cruz, CA 95064, USA*

ARTICLE INFO

Article history:
Received 6 January 2016
Revised 3 July 2016
Accepted 1 August 2016
Available online 2 August 2016

Keywords:
Linear network coding
Information-centric network
Cache management
Bandwidth cost
Cache cost

ABSTRACT

In recent years, the increasing demand for media-rich content has driven many efforts to redesign the Internet architecture. As one of the main candidates, *information-centric network* (ICN) has attracted significant attention, where *in-network cache* is a key component in different ICN architectures. In this paper, we propose a novel framework for optimal cache management in ICNs that jointly considers caching strategy and content routing. Specifically, our framework is based on *software-defined networking* (SDN) where a controller is responsible for determining the optimal caching strategy and content routing via *linear network coding* (LNC). For the proposed cache management framework, we first formulate an optimization problem to minimize the network bandwidth cost and cache cost by jointly considering caching strategy and content routing with LNC. We then develop an efficient *network coding based cache management* (NCCM) algorithm to obtain a near-optimal caching and routing solution for ICNs. We further derive the upper and lower bounds of the problem and conduct extensive experiments to compare the performance of the NCCM algorithm with these bounds. Simulation results validate the effectiveness of the NCCM algorithm and the framework.

1. Introduction

In the past decade, multimedia content has become the dominating traffic over the Internet [2–4]. The increasing demand for media-rich content calls for more efficient methods for content retrieval. To this end, *Information-centric network* (ICN) is a promising design approach that fulfills such a demand by introducing content access by name and enabling *in-network caching* [5,6]. In ICNs, a *content router* (CR) with in-network caching capability can buffer some (usually popular) data chunks for future access [7]. In-network caching can greatly reduce the retrieval delay of content, the traffic in the network, and the service load on the servers [8,9].

To manage in-network caches in ICNs, two major issues need to be jointly considered. One is the *caching strategy* that determines which data chunks shall be cached at each CR, and the other is *content routing* that determines where to route content requests and how to deliver content.

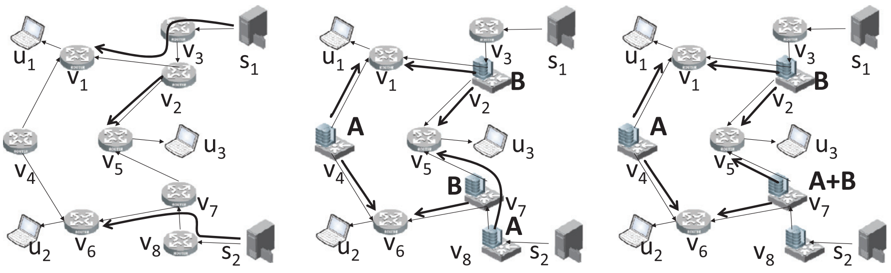
In the literature, there are two types of caching strategies: non-cooperative and cooperative. In non-cooperative caching strategies, a CR opportunistically caches the received data, which may lead to frequent cache updates, sub-optimal cache allocation and caching duplication [8]. In cooperative caching strategies, a CR can collaborate with its neighboring CRs to determine which set of data chunks to cache [9–12].

For content routing, there are two different ways to utilize the in-network caches. One is to only use caches along the path to the original content server for that request and the other is to utilize all nearby caches. The former does not require any cooperation among CRs but may exhibit potentially longer content retrieval delay. The latter requires cooperation among CRs to forward the request to the nearest off-path caches [13]. Either way is closely coupled with content caching. In this paper, we will focus on co-

[☆] A conference version has been published in IEEE/IFIP Networking 2014 [1]. This version contains at least 30% new materials.

[*] Corresponding author.
E-mail address: kejie.lu@upr.edu (K. Lu).

http://dx.doi.org/10.1016/j.comnet.2016.08.004
1389-1286/© 2016 Elsevier B.V. All rights reserved.



(a) With no In-network Cache.

(b) ICNs without LNC.

(c) ICNs with LNC.

Fig. 1. An example of content request in different network scenarios.

operative caching strategy and content routing to fully utilize all distributed in-network caches.

To enable cooperation among distributed CRs, a cache management framework is needed to collect cooperation-related information (e.g., request rates and the current cache status) and make caching and routing decisions. *Software defined networking* (SDN), which physically decouples the control plane and data plane, can satisfy this requirement [14,15]. Typically, on the control plane, a controller is responsible for collecting network information and making routing decisions that will be configured at routers. On the data plane, routers forward packets according to the flow tables configured by the controller. Over the past few years, many new controllers have been designed by using powerful multicore servers to handle a large number of data flows in big networks. For example, McNettle [16] can manage around 20 million requests per second for a network with 5000 switches.

Recently, preliminary studies have been conducted to enable cache management in ICNs based on SDN [17,18]. However, these studies mainly focused on how to incorporate cache related operations into the existing SDN architecture and did not discuss the actual caching strategy. In this paper, we will go one step further to study caching strategy and content routing of ICNs based on SDN with the aim of minimizing both the network bandwidth and cache cost, which is the total cost of bandwidth and cache consumption in the whole network.

Specifically, we will employ *linear network coding* (LNC) to jointly optimize caching strategy and content routing to minimize the network bandwidth and cache cost. We use an example shown in Fig. 1 to illustrate the benefits of using caching and LNC in ICNs. In this figure, a network consists of eight routers (v_1–v_8), and two servers (s_1 and s_2). The users are all connected to routers v_1, v_5, and v_6 and request a piece of content, denoted as f_1, that contains two equal-sized data chunks, *A* and *B*. We assume that each link has a unit cost to transmit one data chunk and a router has a unit cost to cache one data chunk. In terms of the total cost, i.e., the sum of bandwidth cost and cache cost, we have the following results in three different content delivery scenarios:

- In Fig. 1(a), we consider a basic scenario with no in-network cache, so the best way to obtain the designated content is by utilizing multicast with which seven links are used in the routing tree. In this case, there is non in-network cache used. For each data chunk, 7 links will be used and each link has unit capacity. Therefore, to transmit two data chunks, the cache cost is **0** and the bandwidth cost is $2 \times 7 =$**14**. The total cost is $0 + 14 =$**14**.
- In Fig. 1(b), we further assume that there are four CRs (v_2, v_4, v_7 and v_8) and each of them can cache only one data chunk. In this scenario, we consider an ICN without LNC, so each CR can cache one original data chunk. Fig. 1(b) shows the optimal caching strategy and content routing, in which the bold symbol shown on each CR denotes the data chunk cached at the CR. In this case, a total of **4** data chunks are cached in CRs, and transmitting the two data chunks requires **7** units of bandwidth consumption. Therefore, to transmit two data chunks, the cache cost is **4** and the bandwidth cost is **7**. The total cost is $4 + 7 =$**11**, representing a **21**.**42**% improvement.
- Fig. 1(c) shows the scenario with the optimal cache management in ICNs with LNC. In this case, the CRs can cache the linear combination of the original data chunks; and to recover the original data chunks *A* and *B*, a user only needs to obtain any two linearly independent coded data chunks. With the optimal solution, each router (i.e., v_1, v_5 and v_6) can download two coded data chunks from its two nearest CRs, thus CRs only need to cache **3** data chunks and the bandwidth cost is **6** units. Therefore, the total cost is $3 + 6 =$**9**. Compared to the best solution in Scenario 1, the optimal solution for scenario 3 achieves a **35**.**71**% improvement; and compared to the best solution in Scenario 2, it achieves **18**.**18**% improvement.

The above example demonstrates the advantage of jointly considering in-network caching strategy and content routing with LNC in ICNs, which motivates the work of this paper. The main contributions of this paper are summarized as follows.

- We propose a novel SDN-based framework to facilitate the implementation of caching strategy and content routing in ICNs with LNC. The framework is based on the emerging concept of SDN, in which a controller is responsible for determining the optimal caching strategy as well as the optimal content routing via LNC.
- We formulate an optimal cache management problem for ICNs with LNC under a given cache strategy as an *integer linear programming* (ILP) problem. Based on this basic ILP, we further develop the ILP formulation to minimize the total network bandwidth cost and cache cost by jointly considering caching strategy and content routing.
- We develop an efficient *network coding based cache management* (NCCM) algorithm to obtain a near-optimal cache management solution. Based on Lagrangian relaxation, the formulated problem can be relaxed and then decomposed into a linear programming problem and several simple integer maximum weight placement problems, all of which can be solved optimally within polynomial time.
- We conduct extensive experiments to compare the performance of the proposed NCCM algorithm with the lower bound of the ILP formulation. We also compare the performance of the proposed NCCM algorithm with three upper bounds of the problem, i.e., *no cache* (no-Cache), *random cache* (r-Cache) and *greedy cache* (g-Cache) strategies. Simulation results validate the effectiveness of the proposed NCCM algorithm and the framework.

The rest of the paper is organized as follows. We discuss related work in Section 2. In Section 3, we introduce a general cache management framework for ICNs based on SDN. Next, we formulate the optimal cache management problem for ICNs with LNC, which aims to minimize the network bandwidth cost and cache cost by exploiting in-network caches and LNC in Section 4. To solve the problem in practice, in Section 5, we design an efficient algorithm based on Lagrangian relaxation. We then conduct extensive experiments to illustrate the performance of our framework in Section 6. Finally, we discuss the applicability of the proposed scheme in Section 7, and we conclude the paper in Section 8.

2. Related work

It has been shown recently that SDN can significantly improve the cache distribution performance and content management efficiency. In [14] and [15], solutions are proposed and discussed to support ICN by using SDN concepts in wired and wireless networks, respectively. However, there are no discussions on jointly considering the cache managements and routing decisions to further reduce the network cost.

Several cache management systems have been proposed in ICNs [10,12,17,18]. In [17], cache management is integrated with the controller, but the actual caching strategy is left unspecified. In [18], APIs are defined to support cache-related operations, including caching decisions, cache notifications and proactive caching. However, there is no discussion about how to use these APIs to manage caches, nor any concrete caching strategy algorithm is proposed.

There have also been some interests in cooperative caching to improve the cache efficiency in ICNs [9–12]. In [9], CRs on a path are coordinated in a distributed way. The data chunks cached in the downstream CRs are recommended by the upstream CRs. In [10], a distributed cache management system is developed to exchange cache information among all caches to get a global information of the network. Then, cache decisions are made based on the global information of the network by each cache independently. In [11], data chunks are cached based on the chunk number and CRs' label. A complex algorithm is designed to assign CR label for efficiently caching data chunks. These distributed methods may cause complexity and extra overhead to exchange information between CRs. Moreover, there exists a convergence delay for the distributed method. In [12], several centralized off-line replication algorithms are used. Then, an advertisement and request/response mechanism is employed to discover the cached content. Unlike them, we jointly optimize caching strategy and content routing.

Caching strategies can also be found in CDN and web caching [19]. Various models have been studied under different constraints such as link capacity, cache capacity and predicted request demand to minimize average traversed hops, bandwidth consumptions, etc. However, the caching strategy and content routing problems are not jointly optimized for an arbitrary network topology. Moreover, these works are file-based caching strategy and do not employ network coding.

The feasibility and benefits of employing network coding in ICNs are introduced in [20]. It focuses on a distributed approach and some preliminary evaluation results are given. [21] proposes ICNs with built-in network coding, which focuses on signature scheme, communication scheme, forwarding plane and acknowledgement mechanism to optimize Interest forwarding. In [22], we proposed a novel cache-aware K-anycast routing scheme, namely, CAKA, that can significantly improve the performance of content delivery for publish/subscribe-based ICN. Compared to these works, the proposed SDN-based framework is flexible and can facilitate collecting cooperation-related information. It also can enable configuring caching and routing policy in CRs. Furthermore, we also theoretically model the cache management problem in ICNs with LNC, and develop an efficient NCCM algorithm based on Lagrangian relaxation to optimize caching strategy and content routing jointly.

3. A novel cache management framework for ICNs with LNC

In this section, we first introduce the main idea of the cache management framework, which is based on the concept of SDN. We then discuss several important operations to implement the proposed framework.

3.1. The main idea of the framework

In our framework, we consider an ICN that consists of CRs, a controller, and LNC-enabled servers, as shown in Fig. 2. Note that any router can be considered as a CR even if it does not have cache capability. We will regard it as a CR with zero cache capacity. We assume that a content f_n consists of m_n data chunks. We introduce the major functionality related to cache management for ICNs based on SDN as follows.

3.1.1. Functionality of a CR

In our framework, a CR shall be responsible for the following functionality:

- monitoring content requests received from its local end users at the content level (not at the chunk level).
- sending content request statistics to the controller periodically.
- returning data chunks to end users directly if the data chunks are available in its local cache.
- delivering the received data chunks to end users.
- forwarding requests for known content to other CRs according to the flow table.
- forwarding requests for unknown content to the controller.
- retrieving desired coded data chunks from designated servers if necessary.

3.1.2. Functionality of the controller

In our design, the controller shall be responsible for the following functionality:

- gathering content request statistics at the content level (not at the chunk level) from each CR.
- determining Q_k, the local popular content request set[1] for each CR v_k. The routes for content in Q_k needs to be configured ahead of time.
- predicting a popular content set, denoted as *F*, to be cached.
- applying our optimal cache management scheme to optimize the caching strategy and content routing.
- configuring CRs to cache popular content, route content requests and deliver content.
- configuring CRs so that requests for non-popular content can be forwarded to servers.

For a large-scale ICN, the functionality of the controller can also be performed by several cooperated controllers, each of which will be responsible for managing the CRs in its domain and will exchange information with each other.

3.2. Essential operations

In this subsection, we will explain the main operations related to cache management, including how to obtain content request statistics, how to cooperatively cache content in different CRs, where to route content requests and how to deliver content to fully utilize all in-network caches.

[1] In this paper, to reduce the communication overhead, we consider that a local request of a CR is the request sent from the users directly connected with the CR.

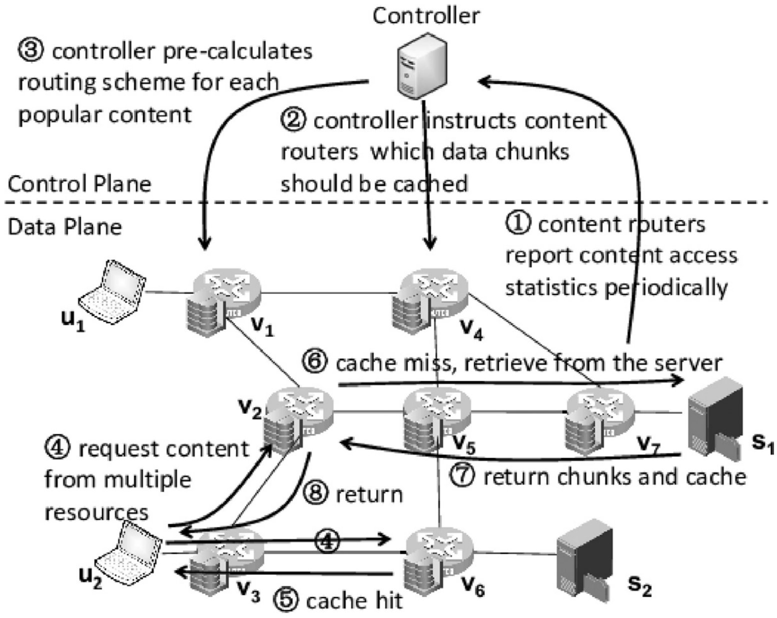

Controller
③ controller pre-calculates routing scheme for each popular content
② controller instructs content routers which data chunks should be cached
Control Plane
Data Plane
① content routers report content access statistics periodically
u1
v1
v4
⑥ cache miss, retrieve from the server
v2
v5
v7
s1
④ request content from multiple resources
⑧ return
⑦ return chunks and cache
u2
v3
④
⑤ cache hit
v6
s2

Fig. 2. A cache management framework.

3.2.1. Content request statistics collection

To obtain optimal caching strategy and routing decisions, all CRs shall report content request statistics periodically with a fixed time period (Step 1 in Fig. 2). Here we note that, with the content request statistics, the controller can adopt any data analysis algorithm to understand the pattern of content request, such as the popularity and the locality over time [23]. Nevertheless, the details about the statistics analysis are out of the scope of this paper.

3.2.2. Caching strategy

Based upon the content request statistics, the controller can determine the local popular content request set (Q_k) for each CR v_k and the set of popular content (F, $F = \bigcup_{v_k} Q_k$). For each content in F, the controller will further determine the set of data chunks that each CR shall cache (Step 2 in Fig. 2). Without using network coding, such periodical and off-line cache placement schemes have been proposed in [12,24]. In practice, the time period can be selected to balance bandwidth cost reduction and the system computation/communication overheads.

To apply LNC, the controller may choose deterministic LNC or random LNC. If deterministic LNC [25–28] is used, the controller shall send a set of *global encoding vectors* (GEVs) to each CR so that the CR can forward a request to LNC-enabled servers to obtain required coded data chunks. In this case, the controller can guarantee that any set of coded data chunks of each content cached in ICNs with size no more than the size of the content is linearly independent. On the other hand, if random LNC [26–29] is used, the controller can send the number of required coded data chunks to each CR, who will then forward the request to LNC-enabled servers that can generate coded data chunks with random GEVs. In such a case, coded data chunks of the same content may be linearly independent to each other with a certain (high) probability. The advantage of the latter scheme is that the computation overhead of the controller can be reduced at the risk of a possible compromise of the linear independence of coded data chunks.

3.2.3. Content routing

In our framework, we classify the content requested by the local end users at each CR into two types, popular and non-popular content. For each CR v_k, the controller should find the optimal route for each popular content in Q_k and configure flow tables in CRs on the route accordingly. The flow entry for a content f_n will have a list of outgoing interfaces, each of which leads to a CR caching coded data chunks of f_n. This step (Step 3 in Fig. 2) can be done after it achieves the optimal caching strategy described in the previous operation.

In our framework, end users only need to obtain any m_n data chunks for content f_n. They can request data chunks one by one as normal. We slightly change the ICN forwarding strategy (e.g., Named Data Networking (NDN)) to make the router's flow entry to record the number of data chunks that can be obtained and have been received from each outgoing interface. When CR v_k receives a request from its local end users for a piece of content in Q_k, it will first check whether it has forwarded another request for the same content with the same sequence number and has not received the returned data chunk. If so, it will not forward this request. If not, the CR will send the request through an outgoing interface if the number of data chunks obtained from this outgoing interface is less than the number of data chunks that can be obtained from this interface (Step 4 in Fig. 2).

If the cache is hit at a CR, the CR will generate a new coded data chunk by randomly combining the cached data chunks of content f_n and send it back (Step 5 in Fig. 2). If the cache is missed at a CR, the CR shall fetch the required data chunks from one or more LNC-enabled servers (Step 6 in Fig. 2) according to the flow table. Once the CR receives the coded data chunks from servers, it will cache the coded data chunks (Step 7 in Fig. 2) and return them to the end users (Step 8 in Fig. 2).

For the request to non-popular content which is not in Q_k, Fig. 3 shows the operation procedure. In particular, when edge CR v_1 receives a request, it will first look up its flow table. If an entry is found, the request will be forwarded accordingly. If an entry cannot be found for the particular content, the edge CR will forward the request to the controller, as shown in Step 2. The controller will then determine an optimal routing scheme and notify all corresponding CRs, which is Step 3. Next, the request may be multicasted to multiple LNC-enabled servers, as shown in Step 4, to obtain the content efficiently using LNC, which is Step 5.

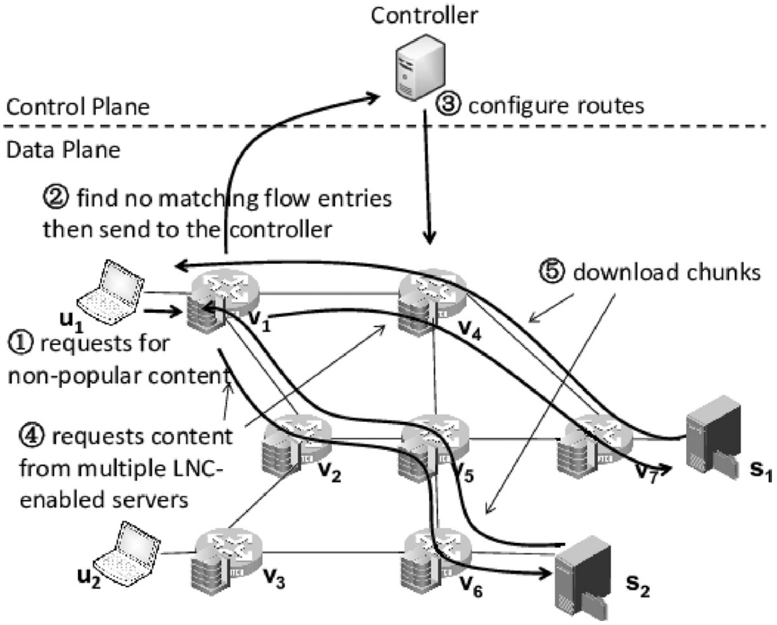

Controller
Control Plane
Data Plane
③ configure routes
② find no matching flow entries then send to the controller
⑤ download chunks
u1
v1
v4
① requests for non-popular content
④ requests content from multiple LNC-enabled servers
v2
v5
v7
s1
u2
v3
v6
s2

Fig. 3. A example of requesting non-popular content.

4. An optimal cache management formulation for ICNs with LNC

In this section, we first state the system model and important assumptions. We then elaborate on the formulation of an optimal problem for cache management in ICNs with LNC.

4.1. The system model

In this paper, we denote an ICN as a directed graph $G =< V, E >$, where V is the set of CRs and E is the set of links between CRs.

4.1.1. Notations

To facilitate further discussions, we summarize the main notations to be used in the paper as follows.

- V: The set of CRs. $V = \{v_1, \ldots, v_{|V|}\}$.
- E: The set of links between CRs. Both $e_{i,\,j}$ and $e_{j,\,i}$ are in E, *iff* there is a link between CR v_i and CR v_j.
- c_k: The cache capacity of CR v_k in terms of data chunks. $c_k \geq 0$, $\forall v_k \in V$.
- $\mathcal{C}_{i,\,j}$: The link cost of link $e_{i,\,j} \in E$ in terms of data chunks. $\mathcal{C}_{i,\,j} \geq 0, \forall e_{i,\,j} \in E$.
- $\mathcal{C}'_{k,n}$: The cache cost when CR v_k caches a data chunk of content f_n.
- F: The set of popular content needed to be cached, $F = \{f_1, \ldots, f_{|F|}\}$.
- m_n: The size of content f_n in terms of data chunks, $\forall f_n \in F$.
- S_n: The set of servers for content f_n, $\forall f_n \in F$.
- Q_k: The set of local popular content requests of CR v_k, $Q_k \subseteq F$, $\forall v_k \in V$.

4.1.2. Assumptions

In our design, we consider that each CR is associated with (1) a cache with a certain capacity, and (2) a set of requests (generated by local end users). As we have explained before, such a general setting can also represent a router with no cache, whose cache capacity is zero. Moreover, we can also use it to represent a server, which not only always stores content it serves but also may have a certain level of cache capacity to cache the content it does not serve.

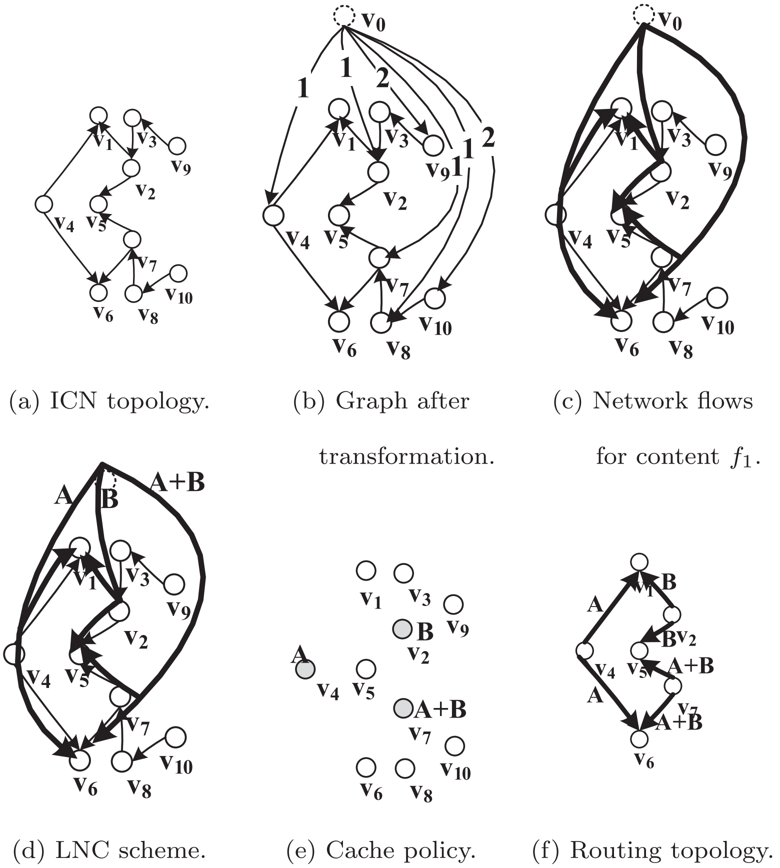
To facilitate the prediction of the popular content set F, as shown in the existing work [30], each CR (which is an open flow switch) can maintain a counter for retrieving the request statistics and the content popularity can thus be inferred directly from these statistics using the measurement module on the controller. The controller can predict the popular content set as follows:

- Each CR v_k periodically counts the number of historical requests for content f_n received from its local end users, which is denoted as $N_{k,n}$.
- Each CR v_k periodically sends the values of $N_{k,n}$ to the controller.
- The controller calculates the requesting popularity of a content f_n at each CR v_k as $p_{k,n} = \frac{N_{k,n}}{\sum_{v_k \in V, f_n \in \bar{F}} N_{k,n}}$, in which $N_{k,n}$ denotes the number of requests for content f_n received by CR v_k from its local end users and $\bar{F}$ denotes the set composed by all contents requested from users. Therefore, we have $\sum_{v_k \in V, f_n \in \bar{F}} p_{k,n} = 1$.
- The *local popular content request set* of each CR v_k is $Q_k = \{f_n | p_{k,n} \geq \overline{p}\}$, in which $\overline{p}$ is a threshold for content popularity[2].
- The controller predicts the *popular content set*, denoted as $F = \bigcup_{v_k \in V} Q_k$. Only the contents in F will be cached in ICN.

Since both the request statistics on each CR and popular content set calculation on the controller are at the content level (not at the chunk level), the above process can be efficiently implemented in the network. Moreover, we note that the controller can adopt any data analysis algorithm to understand the pattern of content request and decide the set of contents which are to be cached in the ICN.

In our design, SDN is used to facilitate cache management and content routing. In SDN, the control plane can be physically decoupled from the data plane, which can be used to obtain content caching and routing decisions by using high level information such as traffic and request statistics. In the literature, many existing studies have implemented ICN based on SDN to provide the ICN functionality using the current Open Flow switches [15,30,31].

[2] The value of $\overline{p}$ determines the number of popular contents which will be cached in CRs.



(a) ICN topology.

(b) Graph after transformation.

(c) Network flows for content f_1.

(d) LNC scheme.

(e) Cache policy.

(f) Routing topology.

Fig. 4. Network flows and cache policy in problem **P**.

In the controller, two modules can be executed for the proposed NCCM scheme. The first one is to calculate the popular content set F. The other is to decide the caching strategy and content routing based on the network status according to the NCCM algorithm. For each CR, it needs to report historical request counts and other status such as available cache capacity to the controller.

When using LNC in ICNs, we assume that each content is firstly divided into data chunks of fixed size[3]. Next, coded data chunks of each content will be generated by the LNC-enabled servers. Consequently, the data chunks cached in the CRs and transmitted in the network are linear combinations of the original data chunks. Since different coded data chunks are linearly independent or independent with high probability, the user only needs to acquire a sufficient number of coded data chunks from any set of CRs to recover the original content. To facilitate further formulation, we assume that each CR requesting content f_n on behalf of its local end users can decode and recover the original data chunks iff it receives no less than m_n coded data chunks of content f_n.

Finally, since most traffic in the network belongs to popular contents downloading, in this paper, we try to optimally cache the coded data chunks of popular contents in the ICN to minimize both the network bandwidth cost and cache cost by jointly optimizing caching strategy and content routing. For the unpopular contents, the controller or edge CRs will route the requests to the original servers according to our framework.

4.2. Problem formulation

Based on the framework and system model we proposed previously, we present an *integer linear programming* (ILP) formulation for the optimal cache management problem. In particular, the objective of the ILP is to minimize the network bandwidth cost and cache cost.

In our framework, CR v_k can download content f_n from multiple CRs. Such a scenario is a typical many-to-one communication for the CR. To formulate the problem, we can add a virtual node v_0 that has all the original data chunks for all pieces of content in F. Moreover, we can add a virtual link with limited capacity from v_0 to each CR $v_d \in V$. We set the traffic load limitation of this virtual link equal to $\beta_{d,\, n}$, *i.e.*, the amount of coded data chunks of content f_n cached at CR v_d. We denote the graph as $\overline{G} = < \overline{V}, \overline{E} >$ after transformation, in which $\overline{V} = V \cup \{v_0\}$ and $\overline{E} = \bigcup_{v_i \in V} \{e_{0,i}\} \cup E$. For the case that CR v_k downloads data chunks of content f_n from multiple CRs in G, it equals to downloading from v_0 in $\overline{G}$. After such transformation, the scenario for many-to-one communication is equivalent to a unicast from v_0 to CR v_k in $\overline{G}$.

Fig. 4(a) and (b) show the original graph G for the example shown in Fig. 1 and the graph $\overline{G}$ generated by transforming graph G. Specifically, the values shown on the links from the virtual node v_0 to other CRs are the traffic load limitation on the links. Since CRs v_2, v_4, v_7 and v_8 have unit cache capacity, the traffic load limitations on these links are one. On the other hand, since we treat server nodes v_9 and v_{10} as CRs which always have all data chunks of the content it serves, the traffic load limitations on these links are two. For CRs v_1, v_3, v_5 and v_6 with no available cache capacity, the traffic load limitations on these links are zero and such links are not shown in Fig. 4(b).

[3] If the available cache capacity of a CR is less than the size of data chunk, it cannot cache any data chunk.

4.2.1. Problem formulation for minimizing network bandwidth cost under a given caching strategy

We first give the problem formulation for a simpler case that the caching strategy is given. Suppose that the values of $\beta_{d,\,n}$ have been decided by a given caching policy. We aim to find the content routing to minimize the cost of network bandwidth while satisfying the content download requirements of end users. To formulate the problem, we give the following definitions:

- $\theta_{k,\,n}$: amount of coded data chunks of content f_n downloaded by CR v_k.
- $\beta_{d,\,n}$: amount of coded data chunks of content f_n cached at CR v_d. $\beta_{d,\,n}$ is a nonnegative integer.
- $l^{i,j}_{k,n}$: traffic load on link $e_{i,\,j}$ for CR v_k downloading content f_n. Specifically, $l^{0,j}_{k,n}$ denotes traffic load from CR v_0 to CR v_j, i.e., the amount of coded data chunks of content f_n downloaded by CR v_k from CR v_j in practice.
- $\ell^{i,j}_n$: bandwidth consumption of traffic on link $e_{i,\,j}$ for downloading content f_n.

Note that a network flow for content f_n means a traffic flow for downloading the coded data chunks of content f_n from different CRs (i.e., downloading from virtual node v_0) to the destination. To transmit content f_n in $\overline{G}$, with the network flow constraints (shown in Eqs. (2)–(4) below) on the unicast for downloading content f_n between v_0 and v_k, the traffic load transmitting on link $e_{0,\,j}$ from v_0 to v_j in $\overline{G}$, i.e., $l^{0,j}_{k,n}$, is equivalent to the amount of coded data chunks of content f_n downloaded from CR v_j by CR v_k in G.

For a given caching strategy, we can formulate the linear programming (LP) as follows:

$$\textbf{Minimize:} \quad \sum_{e_{i,j}\in E}\sum_{f_n\in F}\mathcal{C}_{i,j}\ell^{i,j}_n \tag{1}$$

Subject to:

$$\sum_{v_j\in V} l^{0,j}_{k,n} = \theta_{k,n}, \forall v_k \in V, \forall f_n \in Q_k \tag{2}$$

$$\sum_{j:e_{i,j}\in\overline{E}} l^{i,j}_{k,n} = \sum_{j:e_{j,i}\in\overline{E}} l^{j,i}_{k,n}, \forall v_k \in V, v_i \in \overline{V} - \{v_k, v_0\}, \forall f_n \in Q_k \tag{3}$$

$$\sum_{j:e_{j,k}\in\overline{E}} l^{j,k}_{k,n} = \theta_{k,n}, \forall v_k \in V, \forall f_n \in Q_k \tag{4}$$

$$l^{i,j}_{k,n} \le \ell^{i,j}_n, \forall v_k \in V, \forall e_{i,j} \in E, \forall f_n \in Q_k \tag{5}$$

$$\theta_{k,n} \ge m_n, \forall v_k \in V, \forall f_n \in Q_k \tag{6}$$

$$l^{0,d}_{k,n} \le \beta_{d,n}, \forall v_k, v_d \in V, \forall f_n \in F \tag{7}$$

$$0 \le l^{i,j}_{k,n}, \forall v_k \in V, \forall e_{i,j} \in \overline{E}, \forall f_n \in F \tag{8}$$

In the above formulation, the objective is to minimize the bandwidth cost. Constraints Eqs. (2)–(4) show the network flow constraints on each CR for each content downloading session.

With LNC, if different CRs download different coded data chunks of the same content f_n and the downloaded data chunks should be passing through the same link $e_{i,\,j}$, then the traffic load can be shared by generating and transmitting the random linear combination of the downloaded data chunks. What is more, the new generated coded data chunks passing through link $e_{i,\,j}$ are useful to add new degrees of freedom with high probability to recover all the data chunks of content f_n at each CR requesting it. Thus, only $\max_{v_k\in V} l^{i,j}_{k,n}$ data chunks need to be transmitted through link $e_{i,\,j}$, i.e., the total network bandwidth consumption of traffic on link $e_{i,\,j}$ for downloading f_n is $\max_{v_k\in V} l^{i,j}_{k,n}$. On the other hand, constraint in Eq. (5) equals to:

$$\max_{v_k\in V} l^{i,j}_{k,n} \le \ell^{i,j}_n, \forall e_{i,j} \in E, \forall f_n \in Q_k.$$

Since our objective is to minimize: $\sum_{e_{i,j}\in E}\sum_{f_n\in F}\mathcal{C}_{i,j}\ell^{i,j}_n$, we have $\ell^{i,j}_n = \max_{v_k\in V} l^{i,j}_{k,n}$. Therefore, the objective $\sum_{e_{i,j}\in E}\sum_{f_n\in F}\mathcal{C}_{i,j}\ell^{i,j}_n$ means the total network bandwidth cost.

Constraints in Eqs. (6) and (7) give the content downloading constraints. Specifically, constraint in Eq. (6) means that each CR needs to download sufficient number of coded data chunks to decode and recover the original data chunks for each requested content. Constraint in Eq. (7) shows that the amount of coded data chunks for each content downloaded from a CR is no more than the amount of data chunks of the content cached in the CR. Constraint in Eq. (8) enforce the variables' value range.

After solving the above problem, for each content f_n and each CR $v_k \in V$ requesting f_n, $f_n \in F$, we obtain a unicast network flow with flow capacity no less than m_n between node v_0 and CR v_k. Therefore, a multicast with LNC can be obtained to make sure that each CR requesting content f_n can decode and recover the m_n original data chunks [32].

4.2.2. An ILP formulation for minimize bandwidth cost and cache cost in ICN with NC

When jointly consider the caching strategy and content routing to minimize both the bandwidth cost and cache cost, the parameter $\beta_{d,\,n}$, $\forall v_d \in V$, $f_n \in F$, which denotes amount of coded data chunks of content f_n cached at CR v_d is a nonnegative integer variable.

In the following, we can formulate the problem as an integer linear programming (ILP):

$$\mathbf{P}: \quad \textbf{Minimize:} \quad \sum_{e_{i,j}\in E}\sum_{f_n\in F}\mathcal{C}_{i,j}\ell^{i,j}_n + \sum_{v_d\in V}\sum_{f_n\in F}\mathcal{C}'_{d,n}\beta_{d,n}$$

Subject to:

$$Eq.(2) - Ineq.(8)$$

$$\beta_{d,n} \le m_n, \forall v_d \in V - S_n, \forall f_n \in F \tag{9}$$

$$\beta_{d,n} = m_n, \forall v_d \in S_n, \forall f_n \in F \tag{10}$$

$$\sum_{f_n\in F}\beta_{d,n} \le c_d, \forall v_d \in V \tag{11}$$

$$\beta_{d,n} \in \mathbb{N}, \forall v_d \in V, \forall f_n \in F \tag{12}$$

The first part of the objective denotes the bandwidth cost and the second part is the cache cost. Constraint in Eq. (9) is the cache capacity constraint for each content at each CR, which denotes the number of data chunks cached by each CR is at most the size of the file. Constraint in Eq. (10) shows that the servers for each content always have all the original data chunks for this content. Constraint in Eq. (11) gives the constraint that the amount of data chunks cached at each CR does not exceed its cache capacity. Constraints in Eq. (12) enforce the variables' value range.

In the obtained LNC scheme, the number of coded data chunks transmitted on link $e_{0,\,j}$, $\forall v_j \in V$ in $\overline{G}$ are the number of coded data chunks to be cached at CR v_j in G. In our architecture, if the controller chooses deterministic LNC to implement cache policy, it shall send a set of *global encoding vectors* (GEVs) [25] to each CR, which indicates the coded data chunks the CR should cache. On the other hand, if the controller chooses random LNC, it only sends the number of required coded data chunks to each CR. For each CR $v_i \in V$, the amount of data chunks of content f_n transmitted to its neighboring CR v_j is $\ell_n^{i,j}$. If $\ell_n^{i,j}$ is less than the amount of received data chunks of content f_n, then it generates $\ell_n^{i,j}$ coded data chunks by randomly linearly combining the received data chunks of content f_n and transmits them to v_j. Otherwise[4], it can simply forwards $\ell_n^{i,j}$ coded data chunks it receives to CR v_j.

Fig. 4(c) shows the unicast network flows between v_0 to each CR v_1, v_5 and v_6 for content f_1. Specifically, the traffic on link $e_{0,\,2}$ targeted to CRs v_1 and v_5 is shared; the traffic on link $e_{0,\,4}$ targeted to CRs v_1 and v_6 is shared, and the traffic on link $e_{0,\,7}$ targeted to CRs v_5 and v_6 is shared. According to Fig. 4(c), for the three nodes with available capacity, we have $\beta_{2,1} = 1, \beta_{4,1} = 1$ and $\beta_{7,1} = 1$, which means that each of CRs v_2, v_4 and v_7 caches one coded data chunk for content f_1. Based on the LNC scheme shown in Fig. 4(d), we can obtain the optimal cache policy that data chunk $A + B$ should be cached at CR v_7, data chunk A should be cached at CR v_4, and data chunk B should be cached at CR v_2, which is shown in Fig. 4(e). Moreover, the routing topology shown in Fig. 4(f) can also be acquired according to Fig. 4(c) by removing virtual node v_0 and the links from v_0 to other CRs.

5. An efficient network coding based cache management algorithm

In this section, we design an efficient *network coding based cache management* (NCCM) algorithm to obtain a near-optimal solution of ILP **P** within polynomial computational complexity. Based on Lagrangian relaxation and subgradient algorithm, the NCCM algorithm can be efficiently implemented.

5.1. Lagrangian dual problem

We relax the constraint in Eq. (7) to obtain the Lagrangian dual problem as the resulting problem can be further decomposed into two sub-problems, each of which can be solved within polynomial computational complexity.

Specifically, we first relax the constraint in Eq. (7) by moving it to the objective function with associated Lagrangian multipliers $\lambda_{k,\,d,\,n} \geq 0$, $\forall v_k$, $v_d \in V$, $f_n \in F$. We reformulate the objective function of ILP **P** as follows:

$$\sum_{e_{i,j}\in E}\sum_{f_n\in F}\mathcal{C}_{i,j}\ell_n^{i,j} + \sum_{v_d\in V}\sum_{f_n\in F}\mathcal{C}'_{d,n}\beta_{d,n} + \sum_{v_k,v_d\in V}\sum_{f_n\in F}\lambda_{k,d,n}(l_{k,n}^{0,d} - \beta_{d,n})$$

$$= \sum_{f_n\in F}\left(\sum_{e_{i,j}\in E}\mathcal{C}_{i,j}\ell_n^{i,j} + \sum_{v_k,v_d\in V}\lambda_{k,d,n}l_{k,n}^{0,d}\right) - \sum_{v_d\in V}\sum_{f_n\in F}\left(\sum_{v_k\in V}\lambda_{k,d,n} - \mathcal{C}'_{d,n}\right)\beta_{d,n}.$$

Let $\boldsymbol{\lambda}$ denote the vector composed by elements $\lambda_{k,\,d,\,n}$, $\forall v_k$, $v_d \in V$, $f_n \in F$. Then, the Lagrangian dual problem of ILP **P** is:

$$\max_{\boldsymbol{\lambda}>\mathbf{0}} L(\boldsymbol{\lambda}), \tag{13}$$

in which

$$\mathbf{P}^{\boldsymbol{\lambda}}: \quad L(\boldsymbol{\lambda}) = \min\sum_{f_n\in F}\left(\sum_{e_{i,j}\in E}\mathcal{C}_{i,j}\ell_n^{i,j} + \sum_{v_k,v_d\in V}\lambda_{k,d,n}l_{k,n}^{0,d}\right)$$
$$- \sum_{v_d\in V}\sum_{f_n\in F}\left(\sum_{v_k\in V}\lambda_{k,d,n} - \mathcal{C}'_{d,n}\right)\beta_{d,n}$$

under constraint in Eqs. (2)–(6) and Eqs. (8)–(12).

Obviously, constraints in Eqs. (2)–(6) and Eq. (8) are only related with the group of variables $\{l_{k,n}^{i,j}, \ell_n^{i,j}, \theta_{k,n}\}$ and Eqs. (9)–(12) are only related with the group of variables $\{\beta_{d,\,n}\}$. Therefore, problem $\mathbf{P}^{\boldsymbol{\lambda}}$ can be decomposed into two sub-problems, $\mathbf{P}_1^{\boldsymbol{\lambda}}$ and $\mathbf{P}_2^{\boldsymbol{\lambda}}$ as follows,

$$\mathbf{P}_1^{\boldsymbol{\lambda}}: \ \text{Minimize}: \sum_{f_n\in F}\left(\sum_{e_{i,j}\in E}\mathcal{C}_{i,j}\ell_n^{i,j} + \sum_{v_k,v_d\in V}\lambda_{k,d,n}l_{k,n}^{0,d}\right)$$

under constraints in Eqs. (2)–(6) and Eq. (8). And

$$\mathbf{P}_2^{\boldsymbol{\lambda}}: \ \text{Minimize}: \ -\sum_{v_d\in V}\sum_{f_n\in F}\left(\sum_{v_k\in V}\lambda_{k,d,n} - \mathcal{C}'_{d,n}\right)\beta_{d,n}$$

under constraints in Eqs. (9)–(12).

Firstly, problem $\mathbf{P}_1^{\boldsymbol{\lambda}}$ is a linear programming (LP) problem, which can be efficiently solved. For problem $\mathbf{P}_2^{\boldsymbol{\lambda}}$, we can also design a simple greedy algorithm to obtain its optimal solution, which is shown in Section 5.2.

Given $\boldsymbol{\lambda}$, $B^{\boldsymbol{\lambda}}$ denotes the value of the objective for problem $\mathbf{P}^{\boldsymbol{\lambda}}$ and it is a lower bound for problem **P** [33]. $B^{\boldsymbol{\lambda}}$ can be obtained using the values of the objectives for problem $\mathbf{P}_1^{\boldsymbol{\lambda}}$ and $\mathbf{P}_2^{\boldsymbol{\lambda}}$, denoted as $B_1^{\boldsymbol{\lambda}}$ and $B_2^{\boldsymbol{\lambda}}$, respectively. Thus, $B^{\boldsymbol{\lambda}} = B_1^{\boldsymbol{\lambda}} + B_2^{\boldsymbol{\lambda}}$.

5.2. Optimal algorithm for problem P_2^{λ}

For given $\boldsymbol{\lambda}$, problem $\mathbf{P}_2^{\boldsymbol{\lambda}}$ can be optimally solved as follows. We denote $\lambda^*_{d,n} = \sum_{v_k\in V}\lambda_{k,d,n}$. We have:

$$-\mathbf{P}_2^{\boldsymbol{\lambda}}: \qquad \text{Maximize}: \sum_{v_d\in V}\sum_{f_n\in F}\left(\lambda^*_{d,n} - \mathcal{C}'_{d,n}\right)\beta_{d,n} \tag{14}$$

Subject to: Constraints in Eqs. (9)–(12).

Note that the solution of problem $-\mathbf{P}_2^{\boldsymbol{\lambda}}$ is equivalent to the solution of problem $\mathbf{P}_2^{\boldsymbol{\lambda}}$. However, the value of objective of $-\mathbf{P}_2^{\boldsymbol{\lambda}}$ is $-B_2^{\boldsymbol{\lambda}}$. Problem $-\mathbf{P}_2^{\boldsymbol{\lambda}}$ can be further decomposed into $|V|$ sub-problems. For each CR $v_d \in V$, we only need to solve the following problem.

$$-\mathbf{P}_{2,d}^{\boldsymbol{\lambda}}: \qquad \text{Maximize}: \sum_{f_n\in F}(\lambda^*_{d,n} - \mathcal{C}'_{d,n})\beta_{d,n} \tag{15}$$

Subject to:

$$\beta_{d,n} \leq m_n, \forall f_n \in F, \text{ if } v_d \in V - S_n$$
$$\beta_{d,n} = m_n, \forall f_n \in F, \text{ if } v_d \in S_n$$
$$\sum_{f_n\in F}\beta_{d,n} \leq c_d$$
$$\beta_{d,n} \in \mathbb{N}, \forall f_n \in F$$

The above problem is a maximum weight placement problem with capacity constraints. The optimal algorithm to solve the problem is shown in Algorithm 1. The main idea is that we first sort the set of popular content in decreasing order of their weight $\lambda^*_{d,n} - \mathcal{C}'_{d,n}$. Then, CR v_d caches as many data chunks as possible for each piece of content in the sorted set consecutively. Suppose $B_{2,d}^{\boldsymbol{\lambda}}$ denotes the value of the objective for problem $-\mathbf{P}_{2,d}^{\boldsymbol{\lambda}}$. Then, $B_2^{\boldsymbol{\lambda}} = -\sum_{v_d\in V}B_{2,d}^{\boldsymbol{\lambda}}$.

5.3. Selection of multipliers

To find a good lower bound, the selection of multiplier vector $\boldsymbol{\lambda}$ is important. We use the subgradient optimization to iteratively select $\boldsymbol{\lambda}$. At iteration t, subgradient vector $\boldsymbol{\gamma}^t$ is computed by $\gamma^t_{k,d,n} = l^{0,d,t}_{k,n} - \beta^t_{d,n}, \forall v_k, v_d \in V, \forall f_n \in F$, in which $l^{0,d,t}_{k,n}$ and $\beta^t_{d,n}$ denote the values of variables $l^{0,d}_{k,n}$ and $\beta_{d,\,n}$ in the optimal solution of problem $\mathbf{P}^{\boldsymbol{\lambda}}$ obtained at iteration t. $\boldsymbol{\gamma}^t$ denotes the vector composed by elements $\gamma^t_{k,d,n}, \forall v_k, v_d \in V, f_n \in F$.

[4] In multicast with LNC, the traffic load on each outgoing link of a node is no more than the summation of all the traffic load on its incoming links [32].

Algorithm 1 The greedy algorithm for optimally solving problem $\mathbf{P}^{\boldsymbol{\lambda}}_{2,d}$.

```
For a given node $v_d \in V$ and multiplier vector $\boldsymbol{\lambda}$, compute $\lambda^*_{d,n} = \sum_{v_k \in V} \lambda_{k,d,n}$;
Sort $\lambda^*_{d,n} - \mathcal{C}'_{d,n}$ in decreasing order. Suppose that the $i^{th}$ largest value is $\lambda^*_{d,n_i} - \mathcal{C}'_{d,n_i}$;
Let $\beta_{d,n} = 0, \forall f_n \in F$ and $C = c_d$;
if $\lambda^*_{d,n_1} - \mathcal{C}'_{d,n_1} < 0$ then
  for $i = 1$ to $|F|$ do
    if $v_d \in S_i$ then
      $\beta_{d,i} = m_i$;
      $C$=$C - m_i$;
    end if
  end for
else
  for $i = 1$ to $|F|$ do
    if $v_d \in S_i$ then
      $\beta_{d,i} = m_i$;
      $C$=$C - m_i$;
    end if
  end for
  $j = 1$;
  while $j \le |F|$ and $C \ge 1$ and $\lambda^*_{d,n_j} - \mathcal{C}'_{d,n_j} \ge 0$ do
    if $v_d \notin S_{n_j}$ then
      Cache $\beta_{d,n_j} = \min(C, m_{n_j})$ data chunks of content $f_{n_j}$;
      $C$=$C - \beta_{d,n_j}$;
    end if
    $j = j + 1$;
  end while
end if
return $B^{\boldsymbol{\lambda}}_{2,d}$ and $\beta_{d,n}, \forall f_n \in F$;
```

Multiplier vector $\boldsymbol{\lambda}^{t+1}$ used in the $(t+1)$th iteration can be obtained by $\boldsymbol{\lambda}^{t+1} = \max\{\boldsymbol{\lambda}^t + s_t \boldsymbol{\gamma}^t, 0\}$, in which $\boldsymbol{\lambda}^t$ is the multiplier vector used in the t-th iteration and s_t is a positive step size. s_t can be acquired by a common method [33] as follows: $s_t = \frac{\eta_t (z^t_{UP} - z^t_{LB})}{\|\boldsymbol{\gamma}^t\|^2}$, in which $0 \le \eta_t \le 2$ and $z_{UP}{}^t$ (resp. $z_{LB}{}^t$) denotes an upper (resp. lower) bound on the optimal objective of problem $\mathbf{P}$ in the t-th iteration. Specifically,

$$\eta_t = \begin{cases} 2, & \text{if } t = 1 \\ \eta_{t-1}, & \text{if } z^t_{LB} \ge z^{t-1}_{LB} \\ \eta_{t-1}/2, & \text{if } z^t_{LB} = z^{t-1}_{LB} = z^{t-2}_{LB} \end{cases}. \quad (16)$$

Let $\boldsymbol{\beta}^t$ be the vector composed by elements $\beta^t_{d,n}, \forall v_d \in V, \forall f_n \in F$. To obtain a feasible solution for problem $\mathbf{P}$ at the tth iteration, we let $\boldsymbol{\beta}^t$ be known parameters and solve problem $\mathbf{P}$ with constraints in Eqs. (2)–(7) and Eq. (8). Since the values of $\boldsymbol{\beta}^t$ have already satisfied constraints in Eqs. (9)–(12), the obtained value of the objective for problem $\mathbf{P}$ denoted as $B^{\boldsymbol{\lambda}^t}(\boldsymbol{\beta}^t)$ is an upper bound of the original problem $\mathbf{P}$ obtained at iteration t. Moreover, the obtained solution is obviously a feasible solution of problem $\mathbf{P}$.

5.4. NCCM algorithm based on lagrangian relaxation

We now describe the NCCM algorithm which is designed based on Lagrangian relaxation to solve problem $\mathbf{P}$. Specifically, subproblems $\mathbf{P}^{\boldsymbol{\lambda}}_1$ and $\mathbf{P}^{\boldsymbol{\lambda}}_2$ are solved with multiplier vector $\boldsymbol{\lambda}^t$ at iteration t. At each iteration t, we can obtain a feasible solution for problem $\mathbf{P}$ based on $\boldsymbol{\beta}^t$ and an upper bound of the original problem $\mathbf{P}$. We maintain an upper bound z^t_{UP} as the smallest upper bound we have obtained within t iterations. On the other hand, z^t_{LB} denotes the maximum value of the objective of problem $\mathbf{P}^{\boldsymbol{\lambda}}$ after t iterations, which is a lower bound of problem $\mathbf{P}$.

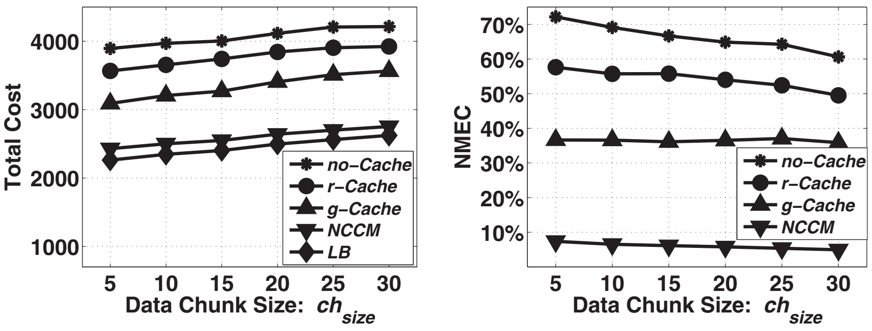
Algorithm 2 Network coding based cache management (NCCM) algorithm.

```
$t = 1$ and $t' = 0$; $z^0_{UP} = +\infty$ and $z^0_{LB} = -\infty$; $\eta_1 = 2$;
Let $\lambda^1_{k,d,n} = 10^{-5}, \forall v_k, v_d \in V, f_n \in F$ and $\epsilon^1 = +\infty$;
while $t < T$ and $\epsilon^t > \epsilon^*$ and $t' < T'$ do
  Solve problem $\mathbf{P}^{\boldsymbol{\lambda}}_1$; Obtain $B^{\boldsymbol{\lambda}^t}_1$ and $l^{0,d,t}_{k,n}, \forall v_k, v_d \in V, \forall f_n \in F$;
  Solve problem $\mathbf{P}^{\boldsymbol{\lambda}}_2$; Obtain $B^{\boldsymbol{\lambda}^t}_2$ and $\beta^t_{d,n}, \forall v_d \in V, \forall f_n \in F$;
  $B^{\boldsymbol{\lambda}^t} = B^{\boldsymbol{\lambda}^t}_1 + B^{\boldsymbol{\lambda}^t}_2$;
  Let $z^t_{UP} = \min(B^{\boldsymbol{\lambda}^t}(\boldsymbol{\beta}^t), z^{t-1}_{UP})$;
  if $z^t_{UP} < z^{t-1}_{UP}$ then
    Let $\pi^*$ be the feasible solution of problem $\mathbf{P}$ obtained at the $t$th iteration in which we let $\boldsymbol{\beta}^t$ be given parameters and solve problem $\mathbf{P}$ with constraints Eq. (2)– Ineq. (7) and Ineq. (8);
  end if
  Let $z^t_{LB} = \max(B^{\boldsymbol{\lambda}^t}, z^{t-1}_{LB})$;
  if $z^t_{LB} > z^{t-1}_{LB}$ then
    $t' = 0$;
  else
    $t' = t' + 1$;
  end if
  if $t' \ge 3$ then
    $\eta_{t+1} = \eta_t/2$;
  end if
  $\epsilon^{t+1} = z^t_{UP} - z^t_{LB}$;
  Update Lagrangian multiplier vector $\boldsymbol{\lambda}^{t+1}$ according to
      $\lambda^{t+1}_{k,d,n} = \max\{\lambda^t_{k,d,n} + s_t \gamma^t_{k,d,n}, 0\}, \forall v_k, v_d \in V,$
  $\forall f_n \in F$, where $\gamma^t_{k,d,n} = l^{0,d,t}_{k,n} - \beta^t_{d,n}$ and $s_t = \frac{\eta_t (z^t_{UP} - z^t_{LB})}{\|\boldsymbol{\gamma}^t\|^2}$;
  $t = t + 1$;
end while
return $z^{t-1}_{UP}$ and $\pi^*$;
```

The NCCM algorithm shown in Algorithm 2 is stopped when one of the conditions is satisfied: (1) the number of iteration t reaches the iteration limit T; (2) the difference between z^t_{LB} and z^t_{UP} is less than a threshold ϵ^*; (3) the lower bound does not increase for more than a number of iterations T'. After the algorithm is terminated, it returns the feasible solution π^* which reaches to the minimum upper bound during the iterations.

6. Numerical results

In this section, we conduct simulation experiments to evaluate the performance of the NCCM algorithm. Specifically, we compare the performance of the NCCM algorithm with a lower bound and three upper bounds. The lower bound, denoted as LB, represents the optimal solution that can be simply obtained by relaxing the integer constraint in Eq. (12) in ILP $\mathbf{P}$. For the first upper bound, we consider a traditional network without using cache and network coding, where data chunks are only available at their original servers. We consider the performance in this case as an upper bound, denoted as *no-Cache*. The other two upper bounds are obtained by (1) randomly caching chunks on all the CRs and (2) greedily caching the contents locally requested for each CR (like LFU cache replacement policy in traditional ICNs without LNC). In these cases, although all the in-network caches are utilized, the potentials of cache management and network coding on the bandwidth cost and cache cost are not fully considered. We denote them as *r-Cache* and *g-Cache*, respectively.



(a) Total cost.

(b) NMEC.

Fig. 5. Cost versus data chunk size.

In our experiments, network G is randomly generated by using a widely used method developed by the Waxman [34]. The method emphasizes the locality of links, which is an important property of many physically deployed network and overlay networks, such as peer-to-peer networks.

Specifically, positions of the nodes, i.e., CRs, in G are a Poisson process in the plane with scaled Lebesgue mean measure, in which η denotes the intensity of the Poisson process. Two CRs v_i and v_j are connected with probability $P(v_i, v_j) = \alpha e^{-d(v_i,v_j)/(\mu * L)}$, where $\alpha > 0$, $\mu \le 1$, $d(v_i, v_j)$ is Euclidean distance between v_i and v_j and L is the maximum distance between any two CRs. Let the region of the random network graph G be 20 × 20. In the experiments shown in Figs. 5–13, we set the network parameters for Waxman network model as follows: the intensity of the Poisson process $\eta \in [0.01, 0.125]$, $\alpha \in [0.5, 1.5]$ and $\mu = 0.7$.

In addition to the network parameters, we have six other parameters in our simulations.

- ch_{size}: the data chunk size, $ch_{size} \in [5\text{ Mb}, 30\text{ Mb}]$.
- ca_{size}: the cache capacity, $ca_{size} \in [50\text{ Mb}, 300\text{ Mb}]$.
- co_{size}: the content size, $co_{size} \in [200\text{ Mb}, 450\text{ Mb}]$.
- $|F|$: the size of the popular content set, $|F| \in [10, 35]$.
- q: the size of the local popular content request set, $q \in [2, 12]$.
- c: cache cost per *Mb*, $c \in [0.01, 5]$.

The bandwidth cost on each link is randomly selected from [0, 1] (per *Mb*). The value of parameter $\mathcal{C}_{i,j}$ (per data chunk) can be calculated by multiplying the link cost (per *Mb*) by the data chunk size. Compared with the network bandwidth cost, the cache cost is lower. The cost for caching each file on each CR is randomly selected from [0, c] (per *Mb*), and the value of parameter $\mathcal{C}'_{d,n}$ (per data chunk) can be calculated by multiplying the cache cost (per *Mb*) and the data chunk size. Since different contents have different sizes and usually only a few nodes in the network are original servers, we set the size of each content in F from [1, co_{size}] and randomly select one CR from the first 10% nodes as its server. For each CR, the cache capacity is set from [0, ca_{size}]. The content request set of each CR is composed by randomly selecting q pieces of content from F.

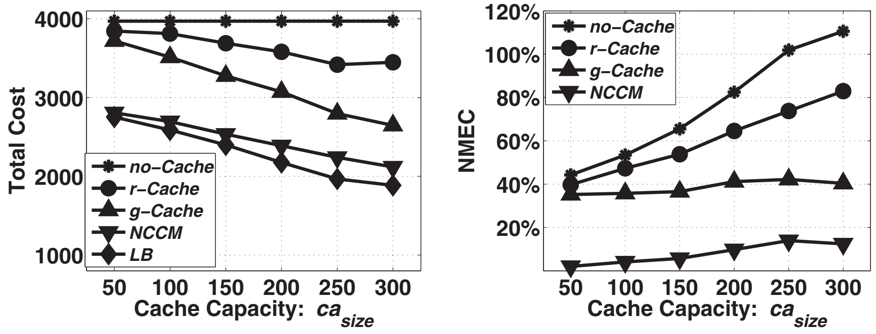
For each simulation instance, we not only compare the total cost, *i.e*, the sum of the network bandwidth cost and the cache cost, obtained by the *NCCM* algorithm with the lower bound *LB* and upper bounds but also show their *normalized maximal extra costs* (NMEC). Specifically, let S_h denote the total cost of *NCCM* (or *no-Cache* or *r-Cache* or *g-Cache*) and S_l denote that of *LB*. The NMEC of *NCCM* (or *no-Cache* or *r-Cache* or *g-Cache*) is defined as $\frac{S_h - S_l}{S_l}$. Unless otherwise specified, we set the number of iterations in *NCCM* $T = 30$. The default values of parameters are: $ch_{size} = 10$, $ca_{size} = 200$, $co_{size} = 300$, $|F| = 20$, $q = 4$, $c = 0.05$, $\eta = 0.025$, $\alpha = 0.7$ and $\mu = 0.7$. In the following discussions, each experiment has one parameter changed from its default value. This helps us to understand how the performance of each strategy is affected by a specific parameter.

In Fig. 5, the total costs of NCCM and all bounds increase slowly with the increase of ch_{size} and the performance of NCCM is very close to the lower bound. The results can be explained as the follows. Firstly, since a data chunk is the data unit transmitted in the network and cached in each CR, the larger the data chunk is, the lower the utilization rate of in-network cache is[5], because the CR cannot cache part of a data chunk. Therefore, utilization rate of in-network cache decreases in this case. Secondly, for each content, it should be divided into a number of data chunks with fixed size. If the size of content cannot be divisible by the size of data chunk, one data chunk should be padded to utilize the network coding. Therefore, the total bandwidth cost to fetch a content for each CR increases in this case. Due to these two reasons, both the cache cost and the bandwidth cost increase with the increase of ch_{size}. For *NCCM*, since the size of each file is fixed, the larger the data chunk is, the fewer data chunks belong to a piece of content, which further reduces the performance of the LNC. Therefore, the decreases of both utilization rate of in-network cache and LNC performance increase the total cost of *NCCM*. In Fig. 5, the total cost of *no-Cache* is the largest, which confirms our expectation that, the utilization of in-network cache reduces the total cost. The total cost of *r-Cache* is higher than the total cost of *g-Cache* which is much higher than the total cost of *NCCM*, which indicates that it is beneficial to jointly consider caching strategy, content routing and network coding to further reduce the total cost. Fig. 5(b) shows the NMEC of upper bounds and *NCCM*. We can clearly observe that the NMEC of *NCCM* is always below 10%.

Fig. 6(a) shows that, the total costs of *r-Cache, g-Cache, NCCM* and *LB* decrease with the increase of ca_{size}. Clearly, when cache capacity of each CR grows larger, more data chunks can be fetched from nearby CRs. Therefore, the network bandwidth cost can be reduced. The total cost *no-Cache* is stable because it does not utilize the in-network cache. Fig. 6(b) shows that the NMEC of *NCCM* first increases and then tends to be stable. On the other hand, the NMECs of three upper bounds keep increasing when ca_{size} grows larger.

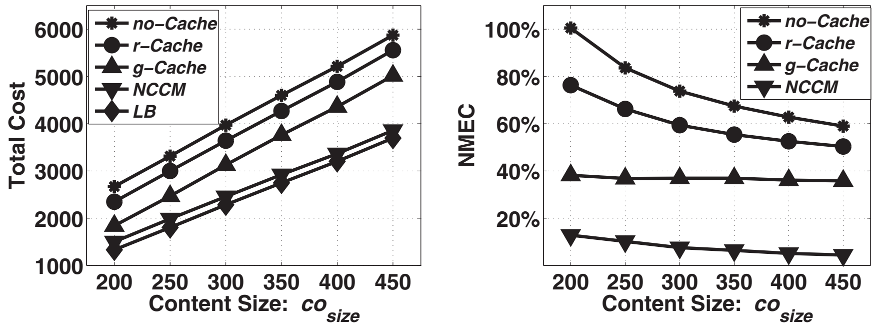
In Fig. 7(a), the total costs of the three upper bounds, *NCCM* and *LB* increase linearly with the increase of co_{size}. Since the number of contents requested by each CR is fixed, the larger the content size

[5] The number of data chunks cached in each CR is an integer. If the available cache capacity on a CR is less than a data chunk, the CR cannot cache the chunk.



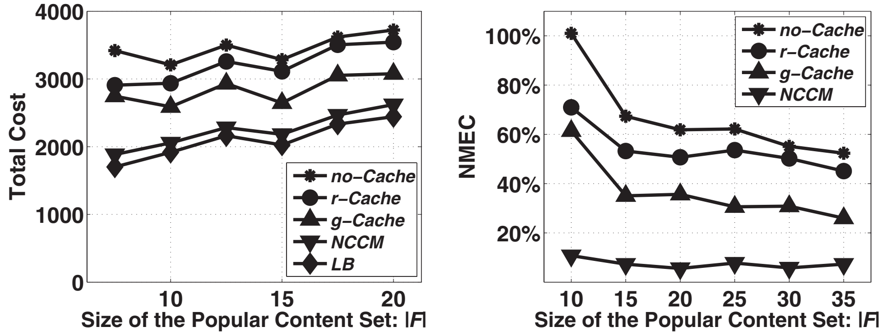
(a) Total cost. (b) NMEC.

Fig. 6. Cost versus data cache capacity.



(a) Total cost. (b) NMEC.

Fig. 7. Cost versus data content size.



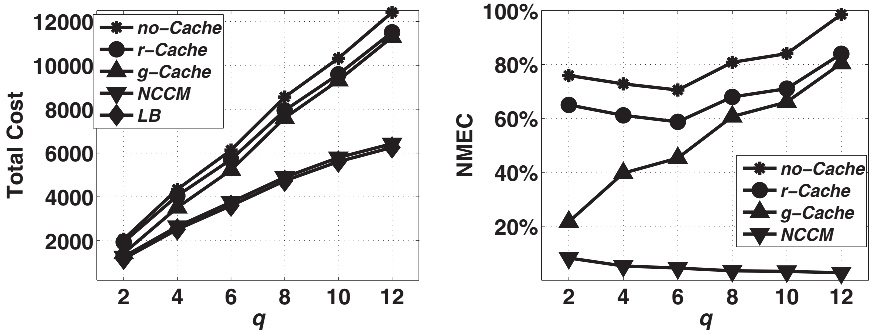
(a) Total cost. (b) NMEC.

Fig. 8. Cost versus number of popular contents.

is, the more bandwidth and cache capacity are consumed. On the other hand, since the cache capacity of each CR is also fixed, when co_{size} increases, the impact of the in-network cache on bandwidth cost reduction becomes smaller. Therefore, as shown in Fig. 7(b), the NMECs of the three upper bounds tend to the same when co_{size} becomes sufficiently large. For *NCCM*, when the co_{size} grows to sufficiently large, each content can be divided into more data chunks, which increases the benefits of the LNC. Therefore, the NMEC of *NCCM* is much smaller than those of the upper bounds.

In Fig. 8(a), when q is fixed, the more popular contents are in F, the fewer CRs request the same content. For each content, the number of chunks cached in all in-network caches decrease. Therefore, the three upper bounds increase slowly. Moreover, the increase of $|F|$ also leads to less coded data chunks transmitted in ICNs with LNC that can be shared between different requesting CRs. Therefore, the total costs of *NCCM* and *LB* increase with the increase of $|F|$. Fig. 8(b) shows that the NMECs of upper bounds are much larger than the NMEC of *NCCM*, and they decrease with increase of $|F|$, because the benefits of the LNC in *NCCM* and *LB* decrease. Nevertheless, the NMEC of *NCCM* is always below 10% and, compared with the upper bounds, it can always save at least 15% total cost.

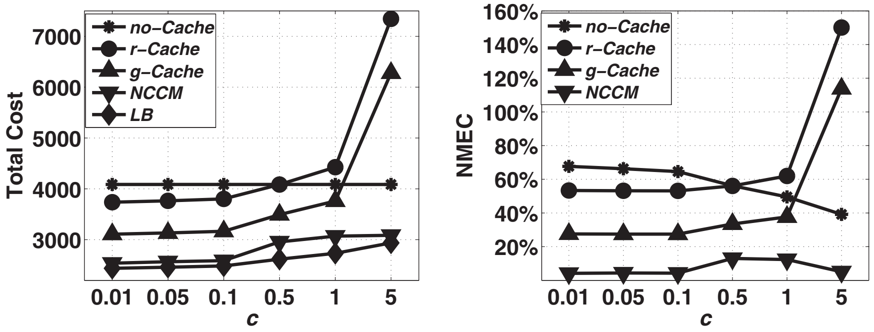
Fig. 9(a) shows that the total costs of the three upper bounds, *NCCM* and *LB* increase with the increase of q. When $|F|$ is fixed, as the number of requested contents increases at each CR, more bandwidth is consumed. On the other hand, since more coded data chunks transmitted in ICNs with LNC can be shared between different requesting CRs, network bandwidth can be reduced by utilizing LNC. The growth rates of *NCCM* and *LB* are lower than those of the upper bounds. Therefore, Fig. 9(b) shows that the NMECs of



(a) Total cost.

(b) NMEC.

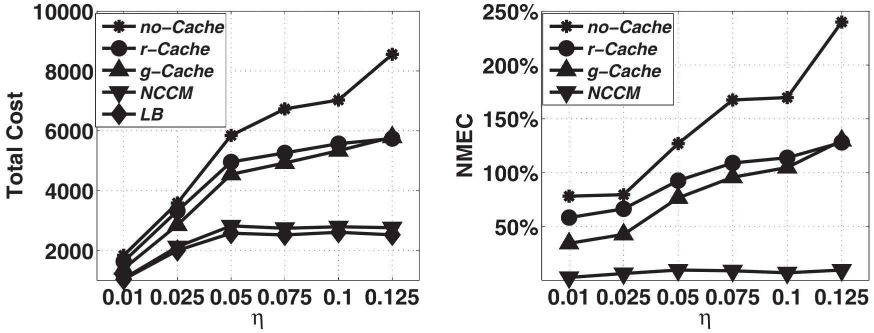
Fig. 9. Cost versus the size of the local popular content request set.



(a) Total cost.

(b) NMEC.

Fig. 10. Cost versus cache cost.



(a) Total cost.

(b) NMEC.

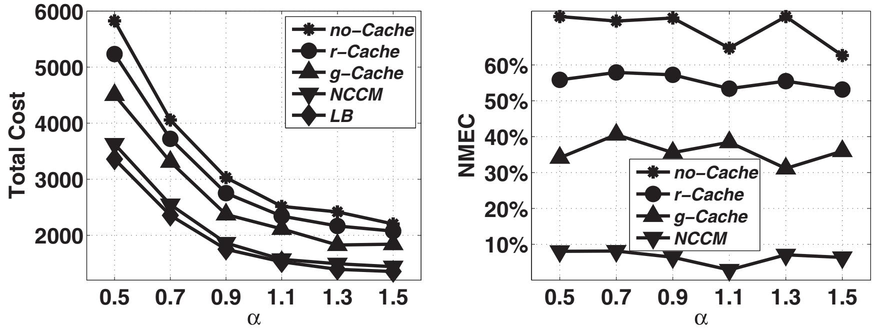
Fig. 11. Cost versus network parameter: η.

the three upper bounds increase with the increase of q. Moreover, we can observe that the NMEC of *NCCM* is decreasing with the increase of q, which demonstrates the advantages of *NCCM*.

Fig. 10(a) shows that the total costs of *r-Cache, g-Cache, NCCM* and *LB* increase with the increase of c. This is reasonable because, when the cache cost per *Mb* grows larger, the total costs increase. The total cost of *no-Cache* keeps stable because it does not utilize the in-network cache. Since *r-Cache* and *g-Cache* always cache data chunks in each CR, the total costs increase with the increase of c and the costs can be larger than that of *no-Cache*. By comparison, since *NCCM* and *LB* jointly consider caching strategy and content routing to minimize the total cost, they cache less data chunks in CRs when c grows larger. Fig. 10(b) shows that the NMEC of *NCCM* is always below 15%.

Fig. 11(a) shows that the total costs of the three upper bounds, *NCCM* and *LB* increase with the increase of η. According to the Waxman model, when η grows larger, there are more CRs in the network. More CRs have more cache capacities. On the other hand, more CRs request more contents, which increases the bandwidth cost. Therefore, in this case, the total cost of the three upper bounds increase. For *NCCM* and *LB*, when the number of CRs increases, more coded data chunks can be shared between different requesting CRs, so network bandwidth can be reduced by utilizing LNC. Therefore, the total costs of *NCCM* and *LB* firstly increase and then tend to be stable. Fig. 11(b) shows that the NMEC of *NCCM* is always below 10% and it can always save at least 30% total cost compared with the upper bounds.

Fig. 12(a) shows that the total costs of the three upper bounds, *NCCM* and *LB* decrease with the increase of α. According to the



(a) Total cost.

(b) NMEC.

Fig. 12. Cost versus network parameter: α.

Waxman model, when α or μ grows larger, since η is fixed, the number of CRs in the network is fixed and the connecting probability of every two CRs increases, which means that the ICN topology is more dense. In this case, the path between a requesting CR and a serving CR will be shorter and more data chunks can be fetched from nearby CRs. Therefore, the network bandwidth cost of *NCCM* or each bound can be reduced. Fig. 12(b) clearly shows that the NMEC of *NCCM* is always below 10% and it can always save at least 20% total cost compared with the upper bounds. Since μ grows larger will also increase more links in the network, simulation results for the cost versus μ have similar trends as those of α.

To summarize, Figs. 5–12 show that the *NCCM* algorithm gets within 10% of a lower bound of the optimal solution under most simulation scenarios.

In our experiments, we also investigate the NMECs and run time of the proposed *NCCM* algorithm vs the number of iterations in Fig. 13. Specifically, when the number of popular contents, $|F|$, and the size of the local popular content request set, q, are small, moderate and large, the results are shown in Fig. 13(a)–(b), (c)–(d) and (e)–(f) respectively. In all these figures, we fix other parameters and vary the values of $|F|$ and q only. In particular, we run the *NCCM* algorithm on a MacBook Pro- 2.8GHz with Intel Core i7. As shown in Fig. 13, the NMEC decreases with the increases of the iteration times and the run time in all the cases. Since the *NCCM* algorithm is designed based on Lagrangian relaxation, the result becomes closer to the optimal solution when the iteration times are set to a larger number. Although the *NCCM* algorithm needs to run iteratively and finally obtain the suboptimal solution, in most cases, the NMEC of a suboptimal solution can converge within four iterations. Fig. 13 also shows that the run time per iteration step becomes larger when the values of $|F|$ and q become larger. However, to achieve a sufficient low NMEC (e.g., less than 3%), the NCCM algorithm only needs 2.37 s even if the number of popular contents, $|F|$ is sufficiently large (e.g., $|F|$=1000 and $q = 20$). Therefore, the proposed *NCCM* algorithm can be efficiently implemented.

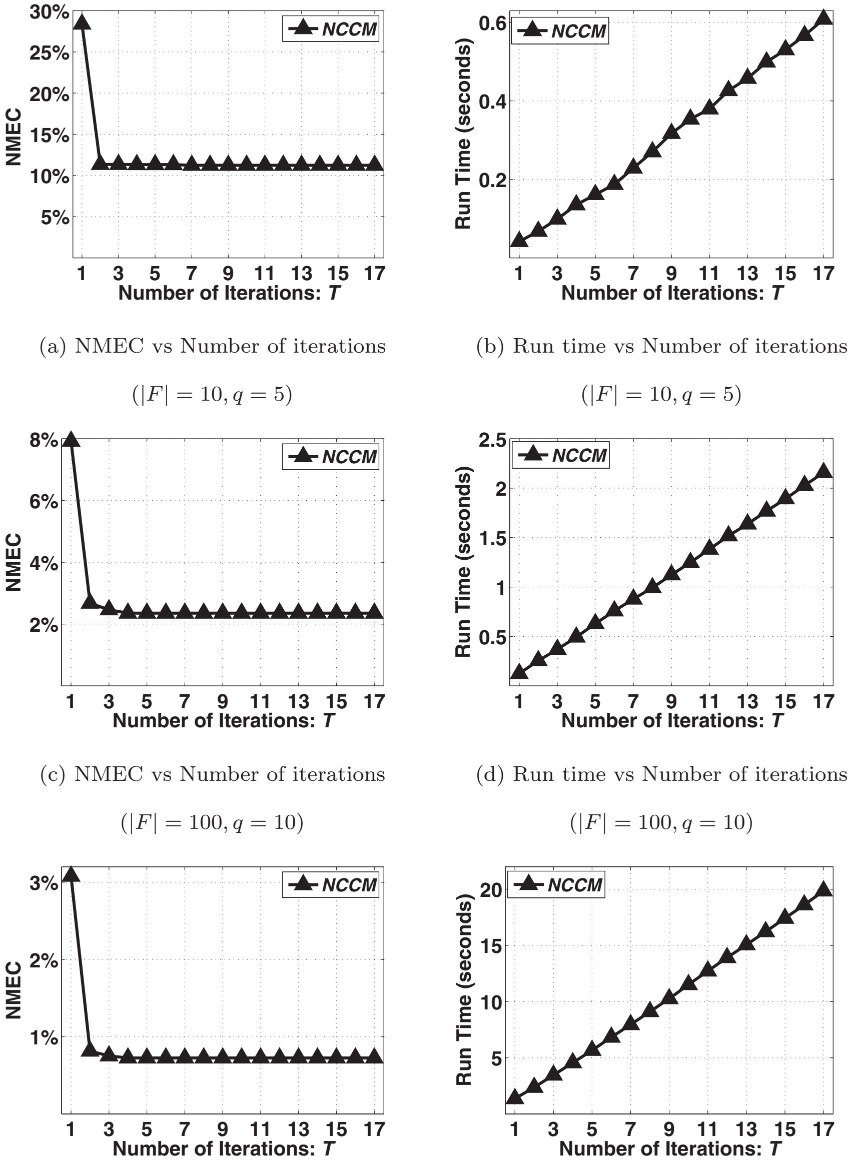
7. Discussions

In this section, we will have discussions on the popularity threshold, the computational overhead of controller, multi-homed devices, failure recovery and mobility of devices.

The threshold determines the number of contents which can be cached in CRs of ICN, i.e., the threshold determines the values of $|F|$. The proposed NCCM algorithm can be used in the general case that the threshold is greater than or equal to zero. If the threshold is zero, then all the contents can be cached in the ICN so the network performance can be optimized but the computational complexity is high. In the case that the threshold is greater than zero, the values of $|F|$ decreases and the number of variables in the proposed NCCM algorithm decreases, which leads to a suboptimal network performance with lower computational complexity. Here we would like to note that, there always exists a tradeoff between the network performance and computational complexity of the proposed NCCM algorithm. Nevertheless, usually, a small number of popular contents ($14\% - 30\%$) consume a large proportion (90%) of bandwidth in traditional networks [35]. Therefore, it is reasonable that we only cache the popular contents in ICN. In practice, the threshold can be determined by considering the network performance and computational complexity of the proposed scheme.

In our design, the main computational complexity is caused by executing the NCCM algorithm. We also believe that the proposed design can be realized in real SDN controller(s) because of the following reasons. Firstly, we note that, in the proposed NCCM scheme, the content caching and content routing for the next time period are pre-calculated based on the historical requests in the current time period. Therefore, the NCCM algorithm is executed off-line, which reduces the requirement of computation capacity of the controller. Secondly, for a large-scale ICN, the functionality of the controller can also be performed by several cooperated controllers, each of which will be responsible for managing the CRs in its domain and will exchange information with each other [36,37]. Thirdly, the computational complexity of the proposed NCCM algorithm is proportional to the number of contents that should be cached in the ICN, i.e., the size of the popular content set F. Since the value of $|F|$ is determined by a threshold for local content popularity, the threshold can be selected to balance the network performance and the computation complexity of the NCCM algorithm. Finally, over the past few years, many new controllers have been designed by using powerful multicore servers to handle a large number of data flows in big networks. For example, McNettle can manage around 20 million requests per second for a network with 5000 switches [16].

To handle the multi-homed devices, firstly, we can introduce a virtual CR with zero cache capacity for each multi-homed device. We assume that the multi-homed device is only connected with the virtual CR and the virtual CR is connected with multiple CRs. All the requests from the multi-homed device can then be treated as the requests of the corresponding virtual CR. The links between each virtual CR and other CRs has zero bandwidth cost. With such a transformation, each multi-homed device can be treated as a single-homed device and we can further obtain a graph G for the input of NCCM algorithm. After we obtain the solution, data chunks transmitted from multiple CRs to a virtual CR means these data chunks are directly transmitted to the multi-



(a) NMEC vs Number of iterations

$(|F| = 10, q = 5)$

(b) Run time vs Number of iterations

$(|F| = 10, q = 5)$

(c) NMEC vs Number of iterations

$(|F| = 100, q = 10)$

(d) Run time vs Number of iterations

$(|F| = 100, q = 10)$

(e) NMEC vs Number of iterations

$(|F| = 1000, q = 20)$

(f) Run time vs Number of iterations

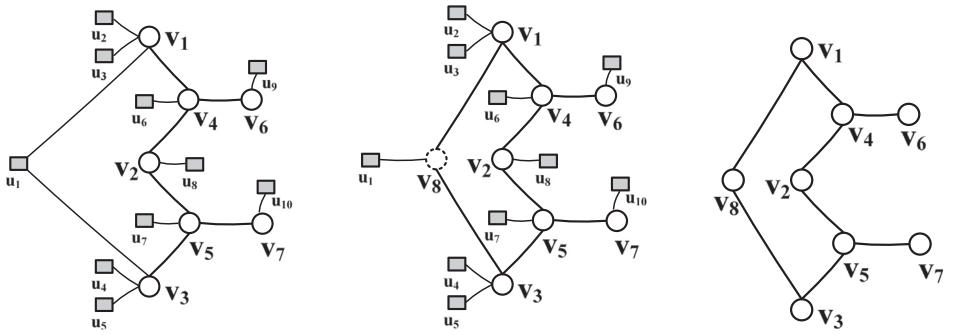
$(|F| = 1000, q = 20)$

Fig. 13. NMEC vs number of iterations (Run time).

homed device corresponding to virtual CR. An example is shown in Fig. 14: As shown in Fig. 14, there are ten devices, which are shown in gray squares, and seven CRs, which are shown in white circles. For the multi-homed device u_1, we add a virtual CR v_8 with zero cache capacity. Then the original network topology can be transformed to a network topology without multi-homed devices. After that we can obtain graph G based on it. We let the bandwidth cost of the links between v_1 and v_8, v_3 and v_8 are zero.

Failure recovery of hardware or protocols is an important issue and has been well studied for SDN [38,39], ICN[40] and SDN based ICN[41], respectively. For instance, [20] shows that, for the data chunk loss due to some failures (such as node failure, link failure, interface lockout, transmission error) and packet drop due to congestion etc., network coding can achieve better performance than the ICN without NC because the coded data chunks retransmitted by CRs are useful for all the devices which still need chunks of the same content. Moreover, the implementation of the ICN with NC has been proposed in [21], in which the failure recovery issue has been handled by using retransmission on packets and retry on alternative paths. Since such hardware or protocols are the basis of SDN and ICN, the proposed NCCM algorithm can be well combined with these existing failure recovery techniques.

In the proposed NCCM scheme, requests are sent from the devices. When a user moves to another place and connects to a new CR, it can simply re-request a number of coded data chunks of a content which have not been decoded and obtained yet. In this case, the user can be treated as a new user and the request only needs to specify the number of coded data chunks, because in ICN with NC, the user can decode and obtain the original content if it can obtain a sufficient number of coded data packets. Therefore,



(a) Original Network Topology. (b) Network Topology Transformation. (c) Graph G.

Fig. 14. An example of network topology transformation for the ICN with multi-homed devices.

this is simpler than that in ICN without NC, such as CCNx. In CCNx, when a user moves to another location, it will re-request all the data chunks that have not been received. It has been shown that CCNx can still handle up to 97% of requests during high mobility [42,43].

8. Conclusion

In this paper, we have systematically explored the power of LNC to reduce the network bandwidth cost and cache cost in ICNs. Specifically, we first proposed a novel SDN based framework for cache management in ICNs with LNC. We then formulated an optimal cache management problem for ICNs with LNC to minimize the network bandwidth cost and cache cost by jointly considering caching strategy and content routing. We also developed an efficient Lagrangian relaxation based algorithm namely, *NCCM*, to obtain near-optimal cache management solution. To evaluate the performance of the framework and the *NCCM* algorithm, we further derived the upper and lower bounds of the problem and conducted extensive experiments to compare the performance of the proposed *NCCM* algorithm with these bounds. Simulation results validate the effectiveness of the proposed *NCCM* algorithm and framework. Specifically, extensive numerical results show that the proposed *NCCM* algorithm gets within 10% of a lower bound of the optimal solution under most simulation scenarios. In addition to minimizing the network bandwidth cost, network security is another key challenge for communication networks. Search schemes over encrypted data have been studied in cloud environment [44–47]. In the future, we will explore the search schemes over encrypted data cached on CRs.

Acknowledgement

This work was supported in part by Collaborative Innovation Center of Novel Software Technology and Industrialization, the Natural Science Foundation of China (No. 61202378, No. 61272462, No. 61271165, No. 61301153, No. 61572310), China's 973 Program (2013CB329103), the Shanghai Oriental Scholar Program, Natural Science Foundation of the Higher Education Institutions of Jiangsu Province No. 16KJB520040, Application Foundation Research of Suzhou No. SYG201401, Hong Kong Research Grant Council under CRF C7036-15G, and Tianjin Key Laboratory of Advanced Networking (TANK), School of Computer Science and Technology, Tianjin University, Tianjin China, 300350.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.comnet.2016.08.004.

References

[1] J. Wang, J. Ren, K. Lu, J. Wang, S. Liu, C. Westphal, An optimal Cache management framework for information-centric networks with network coding, in: IEEE/IFIP Networking Conference, 2014, pp. 1–9, doi:10.1109/IFIPNetworking.2014.6857127.
[2] Cisco Visual Networking Index: Forecast and Methodology, 2014–2019, 2015, Technical Report, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html.
[3] J. Li, X. Li, B. Yang, X. Sun, Segmentation-based image copy-move forgery detection scheme, IEEE Trans. Inf. Forensics Security 10 (2015) 507–518.
[4] Z. Pan, Y. Zhang, S. Kwong, Efficient motion and disparity estimation optimization for low complexity multiview video coding, IEEE Trans. Broadcasting 61 (2015) 166–176.
[5] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, A survey of information-centric networking, IEEE Commun. Mag. 50 (2012) 26–36.
[6] G. Zhang, Y. Li, T. Lin, Caching in information centric networking: A survey, Comput. Netw. 57 (2013) 3128–3141.
[7] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, R.L. Braynard, Networking named content, in: ACMNEXT, December 2009, pp. 1–12.
[8] C. Fricker, P. Robert, J. Roberts, N. Sbihi, Impact of traffic mix on caching performance in a content-centric network, in: IEEE INFOCOM workshop NOMEN'12, March 2012, pp. 310–315.
[9] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, S. Pack, WAVE: popularity-based and collaborative in-network caching for content-oriented networks, in: IEEE INFOCOM workshop NOMEN'12, March 2012, pp. 316–321.
[10] V. Sourlas, L. Gkatzikis, P. Flegkas, L. Tassiulas, Distributed cache management in information-centric networks, IEEE Trans. Netw. Service Manage. Early Access Online (2013).
[11] Z. Li, G. Simon, Time-shifted TV in content centric networks: The case for cooperative in-network caching, IEEE ICC, June 2011.
[12] V. Sourlas, P. Flegkas, G.S. Paschos, D. Katsaros, L. Tassiulas, Storage planning and replica assignment in content-centric publish/subscribe networks, Comput. Netw. 55 (2011) 4021–4032.
[13] M. Draxler, H. Karl, Efficiency of on-path and off-path caching strategies in information centric networks, in: IEEE GreenCom, November 2012, pp. 581–587.
[14] W.-S. Kim, S.-H. Chung, J.-W. Moon, Improved content management for information-centric networking in SDN-based wireless mesh network, Comput. Netw. Available online (2015).
[15] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito, L. Veltri, Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed, Comput. Netw. 57 (2013) 3207–3221.
[16] A. Voellmy, J. Wang, Scalable software defined network controllers, SIGCOMM'12 42 (2012) 289–290.
[17] A. Chanda, C. Westphal, Contentflow: mapping content to flows in software defined networks, IEEE Globecom, 2013.
[18] L. Veltri, G. Morabito, S. Salsano, N. Blefari-Melazzi, A. Detti, Supporting information-centric functionality in software defined networks, in: IEEE ICC, June 2012, pp. 6645–6650.
[19] J. Kangasharju, J. Roberts, K.W. Ross, Object replication strategies in content distribution networks, Comput. Commun. 25 (2002) 376–383.
[20] M.-J. Montpetit, C. Westphal, D. Trossen, Network coding meets information-centric networking, in: ACM MobiHoc workshop NOM'12, June 2012, pp. 31–36.
[21] W.-X. Liu, S.-Z. Yu, G. Tan, J. Cai, Information-centric networking with built-in network coding to achieve multisource transmission at network-layer, Comput. Netw. Available online (2015).
[22] J. Ren, K. Lu, F. Tang, J. Wang, J. Wang, S. Wang, S. Liu, CAKA: a novel cache-aware K-anycast routing scheme for publish/subscribe-based information-centric network, Int. J. Commun. Syst. 28 (2015) 2167–2179.
[23] G. Szabo, B.A. Huberman, Predicting the popularity of online content, Commun. ACM 53 (2010) 80–88.