An Optimal Cache Management Framework for Information-Centric Networks with Network Coding

Jin Wang[1], Jing Ren[2], Kejie Lu[3,4], Jianping Wang[5], Shucheng Liu[6], and Cedric Westphal[6,7]
[1] School of Computer Science and Technology, Soochow University
[2] School of Communication and Information Engineering, University of Electronic Science and Technology of China
[3] College of Computer Science and Technology, Shanghai University of Electronic Power
[4] Department of Electrical and Computer Engineering, University of Puerto Rico at Mayagüez
[5] Department of Computer Science, City University of Hong Kong
[6] Huawei Technologies
[7] Department of Computer Engineering, University of California, Santa Cruz

Abstract—**The increasing demand for media-rich content has driven many efforts to redesign the Internet architecture. As one of the major 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 which jointly considers caching strategy and content routing. Specifically, we propose a cache management framework for ICNs 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). Under the proposed cache management framework, we formally formulate the problem of minimizing the network bandwidth cost by jointly considering caching strategy and content routing with LNC. We develop an efficient *network coding based cache management* (NCCM) algorithm to obtain a near-optimal caching and routing solution for ICNs. We further develop a lower bound of the problem and conduct extensive experiments to compare the performance of the NCCM algorithm with the lower bound. Simulation results validate the effectiveness of the NCCM algorithm and framework.**

I. Introduction

The increasing demand for media-rich content calls for more efficient methods for content retrieval. *Information-centric network* (ICN) is an Internet design approach that fulfills such a demand by introducing named content and enabling *in-network caching* [1]. In ICNs, a *content router* (CR) with in-network caching capability can buffer some (popular) data chunks for future access [2]. In-network caching can greatly reduce the retrieval delay of content, the traffic in the network, and the service load on the servers [3], [4].

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.

This work is supported in part by National Natural Science Foundation of China under Grants No. 61202378, 61272462, Shanghai Oriental Scholar Program, Hong Kong General Research Funding under project CityU No. 9041787, a project from Huawei under CityU No.9231102 and China Post-doctoral Science Foundation No. 2013M531402.

ISBN 978-3-901882-58-6 © 2014 IFIP

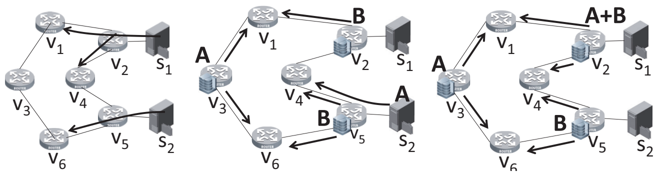
In the literature, there are two types of caching strategies: non-cooperative and cooperative. In non-cooperative caching strategies, a CR opportunistically caches the received data, which may lead to frequent cache updates, sub-optimal cache allocation and caching duplication [3]. In cooperative caching strategies, a CR can sync with its neighboring CRs to determine which set of data chunks to cache [4]–[7].

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 retrieval delay. The latter requires cooperation among CRs to forward the request to the nearest off-path cache [8]. Either way is closely coupled with content routing. In this paper, we will focus on cooperative 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 decouples the control plane and data plane, can satisfy this requirement. Typically, in the control plane, a controller is responsible for collecting network information and making routing decisions which will be configured at routers. In the data plane, routers will forward packets according to the flow tables configured by the controller. Over the past few years, many new controllers has been designed by using powerful multicore servers to scale up to handle a large number of data flows in big networks. For example, McNettle [9] 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 [10], [11]. However, these works 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 the network bandwidth cost, which is the total cost of bandwidth consumption in the whole network.

Specifically, we will employ *linear network coding* (LNC)



(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.

to jointly optimize caching strategy and content routing with the objective of minimizing the network bandwidth cost. We use an example shown in Fig. 1 to illustrate the benefits of using LNC in ICNs. In this figure, a network consists of six routers (v_1–v_6), and two servers (s_1 and s_2). The users are all connected to routers v_1, v_4, and v_6 and request a piece of content, denoted as f_1, that contains two unit data chunks, A and B. We assume that each link has a unit cost. In terms of bandwidth cost, we have the following results in three different network scenarios:

- In Fig. 1(a), no in-network cache has been exploited and the best way to obtain the designated content is by IP multicast where five links are used in the routing tree. To transmit two data chunks, the bandwidth cost is **10**.
- In Fig. 1(b), we further assume that there are three CRs (v_2, v_3, and v_5) and each of them can cache only one data chunk. For ICNs without LNC, each CR can cache one original data chunk. Fig. 1(b) shows the optimal caching strategy and content routing, in which the bold symbols shown on each CR denote the data chunk cached at the CR. In this case, it requires a total of **7** units of bandwidth cost, representing a 30% improvement.
- Fig. 1(c) shows the optimal cache management in ICNs with LNC. The three CRs can cache the linear combination of the original data chunks. 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, *i.e.*, each router (e.g., v_1, v_4 and v_6) can download two coded data chunks from its two nearest CRs, the bandwidth cost is **6**, representing a 40% total 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 the optimal cache management problem for ICNs with LNC to minimize the network bandwidth 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. A near-optimal feasible solution for the cache management problem can be found in a reasonable amount of time.
- We develop a lower bound of the problem and conduct extensive experiments to compare the performance of the proposed NCCM algorithm with the lower bound. Simulation results validate the effectiveness of the proposed NCCM algorithm and framework.

The rest of the paper is organized as follows. In Sec. II, we introduce a general cache management framework for ICNs based on SDN. We formulate the optimal cache management problem for ICNs with LNC, which aims to minimize the network bandwidth cost by exploiting in-network caches and LNC in Sec. III. To solve the problem in practice, in Sec. IV, we design an efficient algorithm based on Lagrangian relaxation. We then conduct extensive experiments to illustrate the performance of our framework in Sec. V. Finally, we discuss related work in Sec. VI and conclude the paper in Sec. VII.

II. 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.

A. 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 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.

1) Functionality of a CR: In our framework, a CR shall be responsible for the following functionality:

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

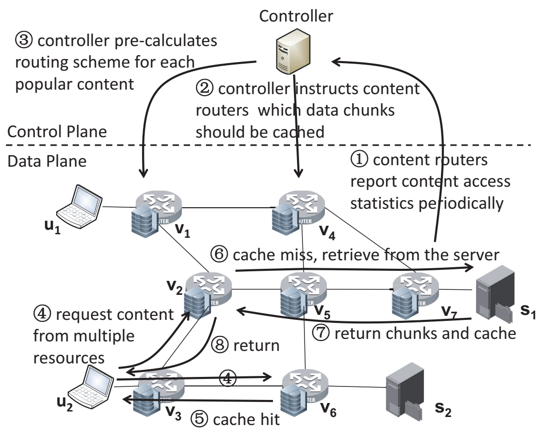

Controller
③ controller pre-calculates routing scheme for each popular content
② controller instructs content routers which data chunks should be cached
Control Plane
Data Plane
① content routers report content access statistics periodically
u1
v1
v4
⑥ cache miss, retrieve from the server
v2
v5
v7
s1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

Fig. 2. A cache management framework.

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 the local popular content request set for each CR v_k, denoted as Q_k, the route of which needs to be pre-configured;
- 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; and
- configuring CRs so that requests for non-popular content can be forwarded to servers.

B. 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) Content Request Statistics Collection: In order to come with 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 [12]. Nevertheless, the details about the statistics analysis are out of the scope of this paper.

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 [7], [13]. 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 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 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) 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 got 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,

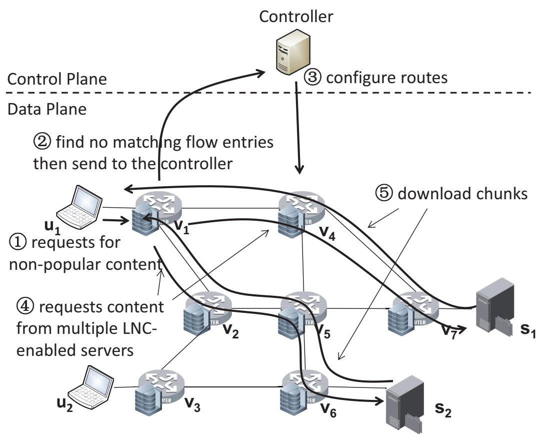

Controller
Control Plane
Data Plane
③ configure routes
② find no matching flow entries then send to the controller
⑤ download chunks
① requests for non-popular content
④ requests content from multiple LNC-enabled servers
u_1
u_2
v_1
v_2
v_3
v_4
v_5
v_6
v_7
s_1
s_2

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

as shown in Step 4, to obtain the content efficiently using LNC, which is Step 5.

III. Optimal Cache Management 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.

A. The System Model

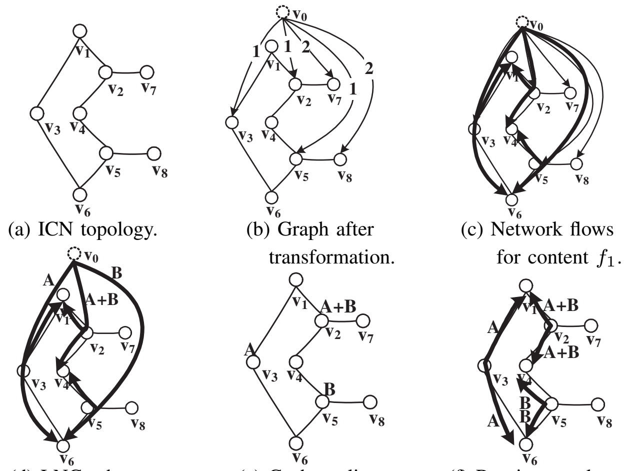
In this paper, we denote an ICN as a directed graph $G =< V, E >$, where V is the set of CRs and E is the set of links between CRs.

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

- V: The set of CRs. $V = \{v_1, \cdots, v_{|V|}\}$.
- 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 capacityof 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. $\mathcal{C}_{i,j} \geq 0, \forall e_{i,j} \in E$.
- F: The set of popular content needed to be cached, $F = \{f_1, \cdots, f_{|F|}\}$.
- 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, $Q_k \subseteq F, \forall v_k \in V$.

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 cache management, we assume that the controller already has the *requesting popularity* of each content f_n at each CR v_k, defined as $p_{k,n}$ ($\sum_{v_k \in V, f_n \in F} p_{k,n} = 1$). Then, it can mark content f_n as a piece of local popular content at CR v_k iff its popularity $p_{k,n}$ is larger than a threshold $\overline{p}$[1]. In other words, the controller already knows the *local popular content request set* $Q_k = \{f_n | p_{k,n} > \overline{p}\}$ for each CR v_k, and the *popular content set* to be cached $F = \bigcup_{v_k \in V} Q_k$.



(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**.

When using LNC in ICNs, we assume that each content is firstly divided into data chunks of fixed size. 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.

B. An ILP 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.

In our framework, CR v_d 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 $\min(m_n, c_d)$. 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 $\overline{G}$.

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

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_3 and v_5 have unit cache capacity, the traffic load limitations on these links are one. On the other hand, since we treat server nodes v_7 and v_8 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_4 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).

We define the following decision variables:

- $\theta_{k,n}$: 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.
- $l_{k,n}^{i,j}$: traffic load on link $e_{i,j}$ for CR v_k downloading content f_n. Specifically, $l_{k,n}^{0,j}$ 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.
- $\ell_n^{i,j}$: 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 Eq. (2)–Eq. (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.*, $l_{k,n}^{0,j}$, is equivalent to the amount of coded data chunks of content f_n downloaded from CR v_j by CR v_k in G.

The ILP formulation **P** to minimize the network bandwidth cost in ICNs with LNC is shown as follows:

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

Subject to:

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

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

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

$$l_{k,n}^{i,j} \le \ell_n^{i,j}, \forall v_k \in V, \forall e_{i,j} \in E, \forall f_n \in Q_k \tag{5}$$

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

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

$$\beta_{d,n} \le m_n, \forall v_d \in V - S_n, \forall f_n \in F \tag{8}$$

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

$$\sum_{f_n\in F} \beta_{d,n} \le c_d, \forall v_d \in V \tag{10}$$

$$\beta_{d,n} \in \mathbb{N}, \forall v_d \in V, \forall f_n \in F \tag{11}$$

$$0 \le l_{k,n}^{i,j}, \forall v_k \in V, \forall e_{i,j} \in \overline{E}, \forall f_n \in F \tag{12}$$

Constraints Eq. (2)–Eq. (4) are 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 $\max\limits_{v_k\in V} l_{k,n}^{i,j}$ 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 $\max\limits_{v_k\in V} l_{k,n}^{i,j}$. On the other hand, constraint Ineq. (5) equals to:

$$\max_{v_k\in V} l_{k,n}^{i,j} \le \ell_n^{i,j}, \forall e_{i,j} \in E, \forall f_n \in Q_k.$$

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

Constraints Ineq. (6) and Ineq. (7) give the content downloading constraints. Specifically, constraint Ineq. (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 Ineq. (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 Ineq. (8) is the cache capacity constraint for each content at each CR. Constraint Eq. (9) shows that the servers for each content always have all the original data chunks for this content. Constraint Ineq. (10) gives the constraint that the amount of data chunks cached at each CR does not exceed its cache capacity. Constraints Ineq. (11) and Ineq. (12) enforce the variables' value range.

After solving the above problem, for each content 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 each CR $v_k \in V$ requesting f_n. 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 [14]. In the obtained LNC scheme, the coded data chunks transmitted on link $e_{0,j}, \forall v_j \in V$ in $\overline{G}$ are the 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 GEVs 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[2], it can simply forwards $\ell_n^{i,j}$ coded data chunks it receives to CR v_j.

[2]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 [14].

Fig. 4(c) shows the unicast network flows between v_0 to each CR v_1, v_4 and v_6 for content f_1. Specifically, the traffic on link $e_{0,2}$ targeted to CRs v_1 and v_4 is shared; the traffic on link $e_{0,3}$ targeted to CRs v_1 and v_6 is shared, and the traffic on link $e_{0,5}$ targeted to CRs v_4 and v_6 is shared. According to Fig. 4(c), for the three nodes with available capacity, we have $\beta_{2,1} = 1, \beta_{3,1} = 1$ and $\beta_{5,1} = 1$, which means that each of CRs v_2, v_3 and v_5 caches one coded data chunk for content f_1. With 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_2, data chunk A should be cached at CR v_3, and data chunk B should be cached at CR v_5, 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. In this example, the total network bandwidth cost is 6.

IV. 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.

A. Lagrangian Dual Problem

We relax the constraint Ineq. (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 Ineq. (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:

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

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

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

under constraints Eq. (2)–Ineq. (6) and Ineq. (8)–Ineq. (12).

Obviously, constraints Eq. (2)–Ineq. (6) and Ineq. (12) are only related with the group of variables $\{l_{k,n}^{i,j}, \ell_n^{i,j}, \theta_{k,n}\}$ and Ineq. (8)–Ineq. (11) 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,

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

under constraints Eq. (2)–Ineq. (6) and Ineq. (12). And

$$\mathbf{P}_2^{\boldsymbol{\lambda}}: \; Minimize: \; -\sum_{v_d\in V}\sum_{f_n\in F}\Big(\sum_{v_k\in V}\lambda_{k,d,n}\Big)\beta_{d,n}$$

under constraints Ineq. (8)–Ineq. (11).

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 Sec. IV-B.

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** [15]. $B^{\boldsymbol{\lambda}}$ 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, $B^{\boldsymbol{\lambda}} = B_1^{\boldsymbol{\lambda}} + B_2^{\boldsymbol{\lambda}}$.

B. Optimal Algorithm for Problem $\mathbf{P}_2^{\boldsymbol{\lambda}}$

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:

$$-\mathbf{P}_2^{\boldsymbol{\lambda}}: \qquad Maximize: \sum_{v_d\in V}\sum_{f_n\in F}\lambda_{d,n}^*\beta_{d,n} \tag{14}$$

Subject to: Constraints Ineq. (8)–Ineq. (11).

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.

$$-\mathbf{P}_{2,d}^{\boldsymbol{\lambda}}: \qquad Maximize: \sum_{f_n\in F}\lambda_{d,n}^*\beta_{d,n} \tag{15}$$

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 $\lambda_{d,n}^*$. 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, $B_2^{\boldsymbol{\lambda}} = -\sum_{v_d\in V}B_{2,d}^{\boldsymbol{\lambda}}$.

C. 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 $\gamma_{k,d,n}^t = l_{k,n}^{0,d,t} - \beta_{d,n}^t, \forall v_k, v_d \in V, \forall f_n \in F$, in which $l_{k,n}^{0,d,t}$ and $\beta_{d,n}^t$ denote the values of variables $l_{k,n}^{0,d}$

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

```
1: For a given node $v_d \in V$ and multiplier vector $\boldsymbol{\lambda}$, compute $\lambda_{d,n}^* = \sum_{v_k \in V} \lambda_{k,d,n}$;
2: Sort $\lambda_{d,n}^*$ in decreasing order. Suppose that the $i^{th}$ largest value is $\lambda_{d,n_i}^*$;
3: Let $\beta_{d,n} = 0, \forall f_n \in F$ and $C = c_d$;
4: for $i = 1$ to $|F|$ do
5:     if $v_d \in S_i$ then
6:         $\beta_{d,i} = m_i$;
7:         $C = C - m_i$;
8:     end if
9: end for
10: $j = 1$;
11: while $C \geq 1$ and $j \leq |F|$ do
12:     if $v_d \notin S_{n_j}$ then
13:         Cache $\beta_{d,n_j} = \min(C, m_{n_j})$ data chunks of content $f_{n_j}$;
14:         $C = C - \beta_{d,n_j}$;
15:     end if
16:     $j = j + 1$;
17: end while
18: return $B_{2,d}^{\boldsymbol{\lambda}}$ and $\beta_{d,n}, \forall f_n \in F$;
```

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 $\gamma_{k,d,n}^t, \forall v_k, v_d \in V, f_n \in F$.

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 [15] as follows: $s_t = \frac{\eta_t(z_{UP} - z_{LB})}{\|\boldsymbol{\gamma}^t\|^2}$, in which $0 \leq \eta_t \leq 2$ and z_{UP} (resp. z_{LB}) denotes an upper (resp. lower) bound on the optimal objective of problem $\mathbf{P}$.

Let $\boldsymbol{\beta}^t$ be the vector composed by elements $\beta_{d,n}^t, \forall v_d \in V, \forall f_n \in F$. To obtain a feasible solution for problem $\mathbf{P}$ at the t^{th} iteration, we let $\boldsymbol{\beta}^t$ be known parameters and solve problem $\mathbf{P}$ with constraints Eq. (2)–Ineq. (7) and Ineq. (12). Since the values of $\boldsymbol{\beta}^t$ have already satisfied constraints Ineq. (8)–Ineq. (11), 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 $\mathbf{P}$.

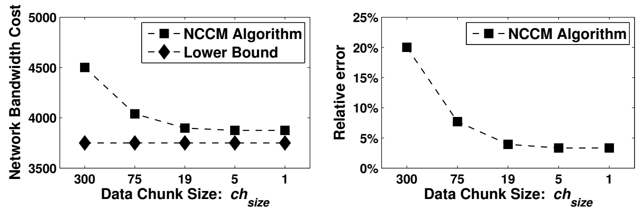
D. NCCM Algorithm based on Lagrangian relaxation

We now describe the NCCM algorithm which is designed based on Lagrangian relaxation to solve problem $\mathbf{P}$. Specifically, sub-problems $\mathbf{P}_1^{\boldsymbol{\lambda}}$ and $\mathbf{P}_2^{\boldsymbol{\lambda}}$ 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_{UP}^t as the smallest upper bound we have obtained within t iterations. On the other hand, z_{LB}^t denotes the maximum value of the objective of problem $\mathbf{P}^{\boldsymbol{\lambda}}$ after t iterations, which is a lower bound of problem $\mathbf{P}$.

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_{LB}^t and z_{UP}^t is less than a threshold ϵ^*; (3) the lower bound does not increase for more than a number of iterations

Algorithm 2 Network Coding based Cache Management (NCCM) Algorithm

```
1: $t = 1$ and $t' = 0$; $z_{UP}^0 = +\infty$ and $z_{LB}^0 = -\infty$; $\eta_1 = 2$;
2: Let $\lambda_{k,d,n}^1 = 10^{-5}, \forall v_k, v_d \in V, f_n \in F$ and $\epsilon^1 = +\infty$;
3: while $t < T$ and $\epsilon^t > \epsilon^*$ and $t' < T'$ do
4:     Solve problem $\mathbf{P}_1^{\boldsymbol{\lambda}}$; Obtain $B_1^{\boldsymbol{\lambda}^t}$ and $l_{k,n}^{0,d,t}, \forall v_k, v_d \in V, \forall f_n \in F$;
5:     Solve problem $\mathbf{P}_2^{\boldsymbol{\lambda}}$; Obtain $B_2^{\boldsymbol{\lambda}^t}$ and $\beta_{d,n}^t, \forall v_d \in V, \forall f_n \in F$;
6:     $B^{\boldsymbol{\lambda}^t} = B_1^{\boldsymbol{\lambda}^t} + B_2^{\boldsymbol{\lambda}^t}$;
7:     Let $z_{UP}^t = \min(B^{\boldsymbol{\lambda}^t}(\boldsymbol{\beta}^t), z_{UP}^{t-1})$;
8:     if $z_{UP}^t < z_{UP}^{t-1}$ then
9:         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. (12);
10:    end if
11:    Let $z_{LB}^t = \max(B^{\boldsymbol{\lambda}^t}, z_{LB}^{t-1})$;
12:    if $z_{LB}^t > z_{LB}^{t-1}$ then
13:        $t' = 0$;
14:    else
15:        $t' = t' + 1$;
16:    end if
17:    if $t' \geq 3$ then
18:        $\eta_{t+1} = \eta_t/2$;
19:    end if
20:    $\epsilon^{t+1} = z_{UP}^t - z_{LB}^t$;
21:    Update Lagrangian multiplier vector $\boldsymbol{\lambda}^{t+1}$ according to
           $\lambda_{k,d,n}^{t+1} = \max\{\lambda_{k,d,n}^t + s_t\gamma_{k,d,n}^t, 0\}, \forall v_k, v_d \in V,$
       $\forall f_n \in F$, where
           $\gamma_{k,d,n}^t = l_{k,n}^{0,d,t} - \beta_{d,n}^t$ and $s_t = \frac{\eta_t(z_{UP}^t - z_{LB}^t)}{\|\boldsymbol{\gamma}^t\|^2}$;
22:    $t = t + 1$;
23: end while
24: return $z_{UP}^{t-1}$ and $\pi^*$;
```



(a) Network bandwidth cost. (b) Relative error.

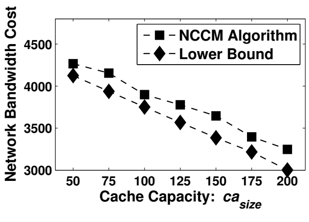
Fig. 5. $|F| = 10, co_{size} = 300, q = 3, ca_{size} = 100$

T'. After the algorithm is terminated, it returns the feasible solution π^* which reaches to the minimum upper bound during the iterations.

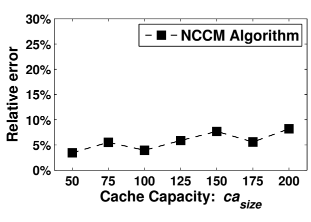
V. Numerical Results

In this section, we conduct simulations to evaluate the performance of the NCCM algorithm. We compare the performance of the NCCM algorithm with a lower bound of the optimal solution that can be simply obtained by relaxing the integer constraint Ineq. (11) in ILP $\mathbf{P}$. In our experiments, we generate a random network graph G by using a widely used method developed by Waxman [16].

We let the region of the random network graph G be 20×20. In the experiments shown in Fig. 5–Fig. 9, we set the

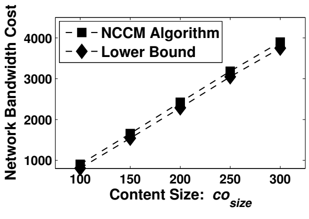


(a) Network bandwidth cost.

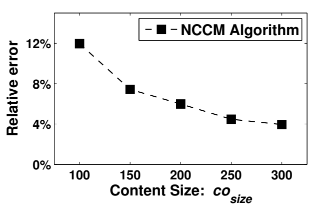


(b) Relative error.

Fig. 6. $|F| = 10, co_{size} = 300, q = 3, ch_{size} = 19$



(a) Network bandwidth cost.



(b) Relative error.

Fig. 7. $|F| = 10, q = 3, ca_{size} = 100, ch_{size} = 19$

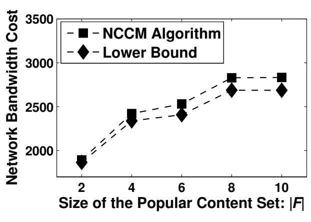
network parameters for Waxman network model as follows: the intensity of the Poisson process $\lambda = 0.025$, $\beta = 0.7$ and $\alpha = 0.7$. Due to space limitations, we do not show the numerical results under different network parameters (λ, β and α).

We have five parameters in our simulations.

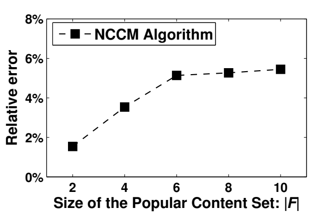
- ch_{size}: the data chunk size, $ch_{size} \in [1Mb, 300Mb]$.
- ca_{size}: the cache capacity, $ca_{size} \in [50Mb, 200Mb]$.
- co_{size}: the content size, $co_{size} \in [100Mb, 300Mb]$.
- $|F|$: the size of the popular content set, $|F| \in [2, 10]$.
- q: the size of the local popular content request set, $q \in [2, 10]$.

The cost on each link is randomly selected from 0.3 to 1 (per Mb). The value of parameter $\mathcal{C}_{i,j}$ (per data chunk) can be calculated by dividing the link cost (per Mb) by the data chunk size. For each content in F, we randomly select one CR as its server. The content request set of each CR is composed by randomly selecting q pieces of content from F. For each instance, we not only compare the network bandwidth cost obtained by the NCCM algorithm with the lower bound, but also show its *relative error*. Let S_h denote the network bandwidth cost of the NCCM algorithm and S_l denote that of the lower bound. The relative error of the NCCM algorithm is defined as $\frac{S_h - S_l}{S_l}$. We set the number of iterations $T = 30$.

In Fig. 5, the network bandwidth cost of the lower bound does not change. When the integer constraint Ineq. (11) is relaxed in ILP **P**, the result of the obtained LP formulation is equivalent to the result of the ILP formulation for the case where ch_{size} is sufficiently small, such that the amount of data for each content cached at each CR obtained by LP formulation is integral multiples of ch_{size}. Therefore, the network bandwidth cost of the lower bound is not affected by the data chunk size. On the other hand, both the network bandwidth cost and the relative error of the NCCM algorithm decrease as ch_{size} gets smaller. The reason is that each content contains more number of data chunks when ch_{size} becomes smaller, which leads to more number of coded data chunks transmitted in ICNs with LNC that can be shared between different requesting CRs. Note that when $ch_{size} = co_{size}$, the case of ICNs with LNC is the same as that without LNC.

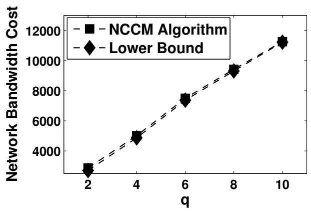


(a) Network bandwidth cost.

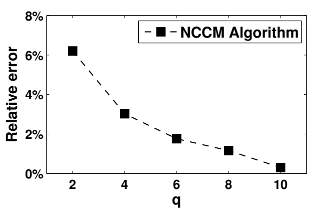


(b) Relative error.

Fig. 8. $co_{size} = 300, q = 2, ca_{size} = 100, ch_{size} = 19$



(a) Network bandwidth cost.



(b) Relative error.

Fig. 9. $|F| = 10, co_{size} = 300, ca_{size} = 100, ch_{size} = 19$

In Fig. 6(a), the network bandwidth costs of both the lower bound and the NCCM algorithm decrease with the increase of ca_{size}. When ca_{size} of each CR grows larger, more data chunks can be fetched from nearby CRs. Therefore, the network bandwidth cost can be reduced. Fig. 6(b) shows that the relative error of the NCCM algorithm is always below 10%.

In Fig. 7(a), the network bandwidth costs of both the lower bound and the NCCM algorithm increase with the increase of co_{size} because the larger the content size, the more bandwidth is consumed. On the other hand, since the effect that co_{size} grows larger is equivalent to that of ch_{size} becoming smaller, as in Fig. 5(b), Fig. 7(b) shows that the relative error of the NCCM algorithm decreases with the increase of co_{size}.

In Fig. 8(a), when q is fixed, the more pieces of content are in F, the fewer CRs request the same content. It leads to less coded data chunks transmitted in ICNs with LNC that can be shared between different requesting CRs. Therefore, the network bandwidth costs of both the lower bound and the NCCM algorithm increase with the increase of $|F|$. The decrease of traffic sharing gives smaller optimization space for the NCCM algorithm. Fig. 8(b) shows that the relative error of the NCCM algorithm increase with the increase of $|F|$.

Fig. 9(a) shows that the network bandwidth costs of both the lower bound and the NCCM algorithm increase with the increase of q. When $|F|$ is fixed, as the number of requested content increases at each CR, more bandwidth is consumed. However, since more coded data chunks transmitted in ICNs with LNC can be shared between different requesting CRs, there is a larger optimization space for the NCCM algorithm. Fig. 9(b) shows that the relative error of the NCCM algorithm decreases with the increase of q.

Fig. 5–Fig. 9 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 simulate the relative errors of the proposed NCCM algorithm vs the number of iterations and run time. In particular, we run the NCCM algorithm on a computer with Intel Core i5-2430 CPU and 4 GBytes memory.

We observe that, although the relative errors decrease with the increases of the iteration times and the run time in all the cases, to achieve a sufficiently low relative error (i.e., less than 3%), the NCCM algorithm only needs no more than 6 seconds even if the number of popular contents, $|F|$ and q are sufficiently large (e.g., $|F|$=1000 and $q = 20$). Therefore, the proposed NCCM algorithm can be efficiently implemented. For a largescale 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.

VI. Related Work

Several cache management systems have been proposed in ICNs [5], [7], [10], [11]. Previous works that are most related to ours are [10] and [11]. In [10], cache management is integrated with the controller, but the actual caching strategy is left unspecified. In [11], 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 has also been some interests in cooperative caching to improve the cache efficiency in ICNs [4]–[7]. In [4], 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 [5], 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 [6], 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 [7], 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 [17], [18]. 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 [19]. Some preliminary evaluation results are given. Compared to that work, we propose a flexible framework to facilitate collecting cooperation-related information. It also can enable configuring caching and routing policy in CRs. We also model the cache management problem in ICNs with LNC, and an efficient NCCM algorithm is designed to optimize caching strategy and content routing jointly.

VII. Conclusion

In this paper, we have explored the power of LNC to reduce the network bandwidth cost in ICNs. We have proposed a novel SDN based framework for cache management in ICNs with LNC. The optimal cache management problem for ICNs with LNC have been considered to minimize the network bandwidth cost by jointly considering caching strategy and content routing. In addition to minimizing the network bandwidth cost, network security is another key challenge for communication networks. In the future, we will explore the potential of LNC to provide the content confidentiality and privacy in ICNs. Moreover, we also will study the impact of network code construction and ratio of popular content on the performance of ICN with LNC.

References

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *ACM CoNEXT*, December 2009, pp. 1–12.

[3] C. Fricker, P. Robert, J. Roberts, and 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.

[4] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, and 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.

[5] V. Sourlas, L. Gkatzikis, P. Flegkas, and L. Tassiulas, "Distributed cache management in information-centric networks," *IEEE Transactions on Network and Service Management*, vol. Early Access Online, 2013.

[6] Z. Li and G. Simon, "Time-shifted TV in content centric networks: The case for cooperative in-network caching," in *IEEE ICC*, June 2011.

[7] V. Sourlas, P. Flegkas, G. S. Paschos, D. Katsaros, and L. Tassiulas, "Storage planning and replica assignment in content-centric publish/subscribe networks," *Computer Networks*, vol. 55, no. 18, pp. 4021–4032, 2011.

[8] M. Draxler and H. Karl, "Efficiency of on-path and off-path caching strategies in information centric networks," in *IEEE GreenCom*, November 2012, pp. 581–587.

[9] A. Voellmy and J. Wang, "Scalable software defined network controllers," *SIGCOMM'12*, vol. 42, no. 4, pp. 289–290, 2012.

[10] A. Chanda and C. Westphal, "ContentFlow: mapping content to flows in software defined networks," in *IEEE Globecom*, Dec. 2013.

[11] L. Veltri, G. Morabito, S. Salsano, N. Blefari-Melazzi, and A. Detti, "Supporting information-centric functionality in software defined networks," in *IEEE ICC*, June 2012, pp. 6645–6650.

[12] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Communications of the ACM*, vol. 53, no. 8, pp. 80–88, 2010.

[13] T. Dirk, "Architecture definition, component descriptions, and requirements," PSIRP project, Tech. Rep. FP7-INFSO-ICT-257217, 2011.

[14] D. Lun, N. Ratnakar, M. Medard, R. Koetter, D. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2608–2623, 2006.

[15] H. D. Sherali and G. Choi, "Recovery of primal solutions when using subgradient optimization methods to solve lagrangian duals of linear programs," *Operations Research Letters*, vol. 19, pp. 105–113, 1996.

[16] B. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[17] E. Cronin, S. Jamin, C. Jin, A. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1369–1382, 2002.

[18] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *IEEE INFOCOM*, March 2012, pp. 2444–2452.

[19] M.-J. Montpetit, C. Westphal, and D. Trossen, "Network coding meets information-centric networking," in *ACM MobiHoc workshop NOM'12*, June 2012, pp. 31–36.