Hindawi
Discrete Dynamics in Nature and Society
Volume 2020, Article ID 3838547, 16 pages
https://doi.org/10.1155/2020/3838547

Research Article

Efficient Coded-Block Delivery and Caching in Information-Centric Networking

Yan Liu[1], **Jun Cai**[1], **Huimin Zhao**[1], **Shunzheng Yu**[2], **JianLiang Ruan**[1] and **Hua Lu**[3]

[1]*Guangdong Polytechnic Normal University, Guangzhou, China*
[2]*School of Data and Computer Science, Sun Yat-San University, Guangzhou, China*
[3]*Network Technology Innovation Center of Guangdong Communication & Network Institute, Guangzhou, China*

Correspondence should be addressed to JianLiang Ruan; ruanjianliang@gpnu.edu.cn

Received 28 March 2020; Accepted 15 May 2020; Published 10 June 2020

Guest Editor: Jianbiao Zhang

Information-centric networking (ICN) provides request aggregation and caching strategies that can improve network performance by reducing content server loads and network traffic. Incorporating network coding into ICN can offer several benefits, but a consumer may receive the same coded block from multiple content routers since the coded block may be cached by any of the content routers on its forwarding path. In this paper, we introduce a request-specific coded-block scheme to avoid linear dependency of blocks that are utilizing in-network caching. Additionally, a non-cooperative coded caching and replacement strategy is designed to guarantee that the cached blocks can be reused. Our experimental results show that the proposed scheme has superior performance to conventional CCN and two network coding-based ICN schemes.

1. Introduction

Trends in recent years have shown that Internet users care more about *what* the content is rather than *where* the content is. Information-centric networking (ICN) [1] is a novel design for a future networking architecture that has been proposed as a promising alternative to the current Internet. In ICN, IP addresses are replaced by content names and content routers (CRs) are equipped with storage capabilities to cache the content passing through each router. Content is requested by Interest packets that are sent by the consumer. With in-network caching [2, 3], the content can be cached by multiple CRs, and any content router (CR) that contains the content that is being requested by the Interest can respond with a data packet, where both the Interest and the data are identified by the content name. Content-centric networking (CCN) has been shown to be a promising ICN architecture [4].

Network coding proposed by Ahlswede et al. [5] has been proven to be helpful in several different network scenarios, including peer-to-peer (P2P) [6], content distribution networks (CDNs) [7], and wireless networks [8, 9]. Recently, several studies have shown that network coding can also offer benefits to ICN [10–21], as network coding can be employed in ICN to effectively utilize multiple paths and reduce the complexity of the cache coordination. However, due to the ICN caching strategy, the same coded block may be cached by multiple CRs on its forwarding path and provided to the same consumer at a later time in response to their multicast requests [22].

In this case, the consumer will not be able to recover the content from the received coded blocks. Several solutions have been proposed to guarantee that all the coded blocks that are provided to the consumer are linearly independent of each other. In some centralized schemes [11, 15], central routers are used to ensure that content caching and routing strategies can provide independent blocks. In some distributed schemes [20, 21], information on the coded blocks which have already been received by the consumer must be carried by the Interest to retrieve linearly independent blocks. The CR can decide whether to respond to the Interest according to the information carried by the Interest. Therefore, several round trips will be required to obtain sufficient linearly independent coded blocks. In our previous

work [23], the CRs only cached the original received blocks to guarantee that all the coded blocks provided to consumers were linearly independent. Any coded blocks generated and transferred were wasted.

To increase the caching efficiency and reduce the cost of computation and communication induced by centralized schemes, we propose a request-specific coded-block (RSCB) scheme to reduce the transmission volume and download delay and ensure that only a single round trip is required for the consumer to retrieve sufficient linearly independent blocks. A non-cooperative coded caching and replacing strategy is then proposed to guarantee that any two coded blocks that are cached in a network will be linearly independent. It is assumed that chunk-based routing and traffic control schemes are in place. The contributions of this paper are as follows:

(i) We propose a special content delivery strategy to retrieve blocks from multiple CRs simultaneously. Each CR on the forwarding path will aggregate Interests received from multiple consumers for chunks of the same content, to eliminate duplicates. Interests received by a CR will be separated again and forwarded in different directions. A mechanism is proposed for the aggregation and separation of Interests for the chunks of content to guarantee that the minimum number of coded blocks will be requested and will be linearly independent.

(ii) An on-path non-cooperative coded caching mechanism is designed to guarantee that the cached blocks can be reused. Blocks received by a CR can be encoded and cached depending on pending Interests and the proposed caching strategy.

(iii) In our model, only chunks (i.e., original blocks) and coded-from-original blocks can be cached. One coded-from-original block can satisfy multiple Interests sent by different consumers requesting a set of its component chunks. A chunk-level coding-instead-of-evicting cache replacement scheme is designed to effectively increase the caching efficiency and optimize cache capacity.

(iv) Our strategy is evaluated by comparison with conventional CCN and two network coding-based ICN strategies. Our experimental results demonstrate that the proposed strategy achieves the highest performance in terms of parameters such as average download time, server hit reduction rate, and cache hit rate.

2. Related Works

Network coding techniques have received much attention in a variety of network scenarios including P2P networks [6], CDNs [7], and wireless networks [9]. Recently, several works have been proposed that apply network coding in ICN. There are two categories of solutions that can be used to ensure consumers are provided with sufficient linearly independent coded blocks: centralized strategies and distributed strategies.

Wang et al. [24] proposed a novel SDN-based framework to implement content caching and routing in ICN with linear network coding. The SDN controllers determine how to cache and route based on the information collected by the CR. Thus, a near-optimal caching and routing strategy can be obtained. Sadjadpour [11] proposed an architecture based on index coding for ICN, which groups the nodes into several clusters. The central router of each cluster maintains information on which content is cached by each node. Coded blocks generated by the central router are used to satisfy Interests for different content sent by different nodes. However, this strategy does not reduce traffic the first time content is requested. Llorca et al. [14] presented a multicast scheme based on network coding to achieve maximum network efficiency. However, the proposal does not mention a solution to deploy the proposed strategy in ICN. Talebifard et al. [15] proposed a method based on network coding that reduces the costs of coding and decoding by breaking the network into several clusters, with network coding only performed by selected nodes or clusters.

As well as centralized strategies based on network coding, some works have obtained enough linearly independent coded blocks by sending Interests repeatedly. Zhang and Xu [21] proposed two checking strategies to guarantee that the consumer will receive sufficient linearly independent coded blocks, which were called precise matching and RB matching. In precise matching, each Interest carries the global coefficients X of the coded blocks that have already been received by the consumer. Each CR performs Gaussian elimination to check linear dependencies. Precise matching is an efficient approach to guarantee that all blocks received by consumers will be linearly independent. However, it has very high communication and computation overheads. Therefore, RB matching was proposed as a more lightweight approach, where the Interest only carries the rank of the global coefficients X of the coded blocks already received by the consumer. If the number of coded blocks cached by the CR is larger than the rank of the global coefficients X, the CR can respond to the Interest with a coded block. The larger the value of $|X|$ is, the more difficult it is to serve the Interest. Wu et al. [16] proposed a network coding and random forwarding-based caching strategy, CodingCache, to enhance the caching efficiency. To guarantee all the blocks provided to consumer are linearly independent, each Interest carries the global coefficients of the coded blocks already received to retrieve the next block, similar to precise matching. Therefore, N rounds will be required in order to retrieve N blocks. Nguyen et al. [20] proposed a lightweight caching and Interest aggregation strategy to ensure that all the coded blocks received by the consumer are independent. Like RB matching, the rank of the global coefficients of the coded blocks already received by the consumer is carried by the Interest packet. Saltarin et al. [19] proposed a protocol named NetCodCCN to permit Interest aggregation and pipelining. Each node responds to an Interest once it has received enough coded blocks to recover the content or $|X|$ is larger than the number of coded blocks already sent out over face i previously, where $|X|$ is the rank of the global coefficients X of the coded blocks cached in ContentStore.

However, NetCodCCN has a weakness also shared by RB matching in that it may provide false negative decisions, i.e., a node may falsely decide it cannot provide an innovative coded block for the consumer while actually the block is available. Montpetit et al. [17] proposed an architecture based on network coding, NC3N, where each Interest retrieves one coded block. However, there method does not include a strategy to ensure all the received blocks are independent. Liu et al. [18] proposed an ICN-NC method to guarantee that all the blocks received are provided by different CRs to increase the probability of obtaining linearly independent blocks. Each Interest packet contains a record of the Interest exploration range of the previous round. Only CRs within a new exploration area are permitted to respond to these Interests. Several rounds are required to retrieve enough independent coded blocks, and the Interest may retrieve linearly dependent coded blocks. The authors in [25] proposed a framework based on network coding for cache management in ICNs. Saltarin et al. [26] proposed a distributed caching strategy for ICNs enabled for network coding, which gives CRs the responsibility of estimating the popularity of contents and ensuring that the most popular content is cached near the network edge.

Most of the existing schemes require several round trips to obtain sufficient linearly independent coded blocks to recover the content. In this paper, we propose a novel content delivery strategy to ensure enough blocks can be retrieved within a single round. An on-path non-cooperative caching and replacing strategy based on network coding is proposed to guarantee that all blocks received by consumers are linearly independent. Moreover, in our scheme, coded blocks are generated only if the traffic can be saved instead of generated at the server and all CRs on the forwarding path in order to reduce the cost of coding and decoding.

3. Method of Interest Aggregation and Separation

In ICN, chunk-based delivery strategies route chunks separately. Chunks may meet on an intermediate node during their forwarding paths to several consumers. Motivated by this, we propose a special request-specific coded-block (RSCB) scheme to encode chunks that meet during transport in order to reduce traffic.

3.1. Overview of RSNC. The definitions given in our previous study (referred to as RSNC) will be followed here. Each Interest (S, N) requests a specific set of chunks, where $S = \{1, \ldots, N\}$ is the set of chunk indices and N is number of independent coded blocks required to recover the content. Since chunks may be cached by different CRs, each CR can aggregate, separate, and forward Interests. If Interest 1 requests a set of chunks S_1 and Interest 2 requests a set of chunks S_2, then (S, n) satisfies both Interests, where $S = S_1 \cup S_2$ is the set of chunks used to generate n linearly independent coded blocks, n is the number of coded blocks to be sent by the upstream CR, and $n = \max\{|S_1|, |S_2|\}$. When $n < |S|$, the traffic required to deliver chunks from upstream will be reduced.

Since an Interest sent by a consumer for multiple chunks will be copied and forwarded along a multicast tree, requests for different chunks sent from the same consumer will not meet again in a CR on the multicast tree. Therefore, the Interest aggregation operation "$\oplus$" is defined to combine two Interests originating from at least two different consumers, i.e.,

$$\begin{aligned} &(S_1, n_1) \oplus (S_2, n_2) \overset{\text{def}}{\Rightarrow} (S, n), \\ &\text{where } S = S_1 \cup S_2, \\ &n = \max\{n_1, n_2\}. \end{aligned} \tag{1}$$

Similarly, a separation operation "Div" is defined to split an Interest into several sub-Interests:

$$\begin{aligned} &\text{Div}(S, n) \overset{\text{def}}{\Rightarrow} \{(S_3, n_3), (S_4, n_4)\}, \\ &\text{where } (S_3, n_3) \oplus (S_4, n_4) = (S, n), \\ &S_3 \cap S_4 = \varnothing, \\ &S = S_3 \cup S_4, \\ &n_3 = \min\{|S_3|, n\}, \\ &n_4 = \min\{|S_4|, n\}. \end{aligned} \tag{2}$$

3.2. Interest Aggregation and Separation in RSCB. In contrast with RSNC [23], RSCB includes information on (S_1, n_1) and (S_2, n_2) in the aggregated Interest (S, n) which guarantees that linearly independent coded blocks are provided to consumers and minimize the number of coded blocks transported in the network. To reduce the size of the Interest, the sub-Interest information is presented as a binary number, $b(S_i)$. For example, if Interest $(S = \{1, 2, 3, 4\}, b(S), n = 2)$ is an aggregated Interest of Interest 1 $(S_1 = \{1, 3\}, n_1 = 2)$ and Interest 2 $(S_2 = \{2, 4\}, n_2 = 2)$, the binary information of Interest 1 is $b(S_1) = 1010$, and the binary information of Interest 2 is $b(S_2) = 0101$, and thus $b(S) = b(S_1) \cup b(S_2) = \{1010, 0101\}$. Therefore, an Interest can be expressed as $I(p, [S, b(S)], n)$, where p is the name of the requested content, S is a set of chunks that match the name of the content p, $b(S)$ is a set of binary numbers representing the sub-Interests (each sub-Interest is a subset of S), and n is the number of linearly independent coded blocks being requested. Therefore, any n linearly independent coded blocks that contain all chunks specified by S will satisfy the sub-Interests specified in $b(S)$.

Equation (1) can thus be further modified as

$$\begin{aligned} &([S_1, b(S_1)], s_1) \oplus ([S_2, b(S_2)], s_2) \overset{\text{def}}{=} ([S, b(S)], s), \\ &\text{where } S = S_1 \cup S_2, \\ &s = \max\{s_1, s_2\}, \\ &b(S) \subseteq b(S_1) \cup b(S_2), \end{aligned} \tag{3}$$

where s is the minimum number of linearly coded blocks satisfying both Interests $([S_1, b(S_1)], s_1)$ and $([S_2, b(S_2)], s_2)$.

It should be noted that the binary number $b(S_i)$ is used to represent the subset information, which is required to guarantee that the requested number of coded blocks is minimized. When $s = 1$ or $s = |S|$, this subset information is not necessary, as shown in Figure 1(a). Moreover, if $S_i \subseteq S_j$, the information on S_i, i.e., $b(S_i)$, will be deleted from $b(S)$.

For instance, Figure 1(a) shows that CR_1 receives two Interests for content C_p from different interfaces, $I(p, S_1 = \{2, 4\}, 2)$ and $I(p, S_2 = \{1, 3\}, 2)$. Before these two Interests are forwarded, CR_1 aggregates the two requests into a single Interest $I(p, [S = \{1, 2, 3, 4\}, b(S) = \{0101, 1010\}], 2]$ using equation (3). $b(S)$ can then be used to reconstruct the subsets $\{2, 4\}$ and $\{1, 3\}$.

Similarly, the separation operation used to split an Interest $([S, b(S)], s)$ into several sub-Interests is modified, which is used to distribute the sub-Interests out over several interfaces of the CR towards different content sources:

$$\begin{aligned}
&\mathrm{Div}([S, b(S)], s) \stackrel{\mathrm{def}}{=} \{([S_1, b(S_1)], s_1), ([S_2, b(S_2)], s_2)\},\\
&\text{where } S_1 \cap S_2 = \text{null},\\
&S = S_1 \cup S_2;\\
&s_1 = \min\{|S_1|, s\},\\
&s_2 = \min\{|S_2|, s\}.
\end{aligned} \tag{4}$$

If an Interest $([S, b(S)], s)$ is formed by merging multiple Interests, the subsets (S_i, s_i) should be reconstructed based on $b(S)$, and these subsets should then be separated into sub-subsets using equation (2). Then, new Interests, i.e., $([S_1, b(S_1)], s_1)$ and $([S_2, b(S_2)], s_2)$ in equation (4), are generated by aggregating the sub-subsets using equation (3). This procedure is described in Algorithm 1. The complexity of Algorithm 1 is $O(n)$, where n is the number of subsets. According to Algorithm 1, $I(p, [\{1, 2, 3, 4\}, b(S)], 2)$ can be reconstructed into two subsets $(\{2, 4\}, 2)$ and $(\{1, 3\}, 2)$ for $b(S) = \{0101, 1010\}$. These subsets can be further divided into sub-subsets: $(\{2\}, 1)$, $(\{4\}, 1)$, $(\{1\}, 1)$, $(\{3\}, 1)$ using equation (2), and then aggregated into Interest $I(p, \{1, 2\}, 1)$ and Interest $I(p, \{3, 4\}, 1)$ according to equation (3). If the original blocks ob_1 and ob_2 are located in the same direction and the original blocks ob_3 and ob_4 are located in another direction, the new Interests can be sent from two interfaces in two different directions, as shown in Figure 1(a). In this case, only two blocks will transmitted which is in contrast with RSNC [23], which requires four blocks to be transmitted.

If Interest 2 (S_2, n_2) arrives after Interest 1 (S_1, n_1) has already been sent upstream, then the aggregated pending Interest will be $(S, n) \stackrel{\mathrm{def}}{=} (S_1, n_1) \oplus (S_2, n_2)$. Since (S_2, n_2) may contain some chunks that have also been requested by Interest (S_1, n_1), these chunks should be removed from Interest 2. Therefore, we define an operation to determine incremental Interest based on the separation operation:

$$\begin{aligned}
&(\Delta S_2, \Delta n_2) \stackrel{\mathrm{def}}{=} (S_2, n_2) \setminus (S_1, n_1),\\
&\text{where } \Delta S_2 = S_2 \setminus \Delta S_1, \Delta n_2 = \min\{|\Delta S_2|, n_2\},\\
&\Delta S_1 \subseteq S_1 \cap S_2,\\
&\Delta n_1 = \min\{|\Delta S_1|, n_1\}.
\end{aligned} \tag{5}$$

Since Interest 1 will return at most n_1 coded blocks, $\Delta n_1 \le n_1$ is required. If $|S_1 \cap S_2| > n_1$ and $n_2 > n_1$, we let $\Delta S_1 \subset S_1 \cap S_2$ and $\Delta n_1 = |\Delta S_1| = n_1$. Similarly, if a CR has cached a subset (W, w) of blocks, only the remaining (S', n') blocks need to be requested from the upstream CRs, where $(S', n') = (S, n) \setminus (W, w)$. In RSCB, the coded blocks generated by the original blocks (referred to as *coded-from-original block*) are cached by the first node *en route* to consumers. The coded-from-original blocks are used as the original blocks to satisfy future Interests. For example, in Figure 1(b), the coded-from-original block, $ocb_1 = \alpha_{11} ob_1 + \alpha_{12} ob_2$, can be presented as $(W = \{1, 2\}, w = 1)$. When Interest $(S = \{1, 2\}, n = 2)$ is received by CR_2, the incremental Interest $(S' = 2, n = 1)$ is determined and sent to the next node.

The benefits of RSCB are illustrated in Figures 1 and 2. In Figure 1(a), two consumers connected to router CR_5 and CR_6 have requested content, which contains four original blocks, ob_1, ob_2, ob_3, and ob_4 at the same time. Each original block has a size of one unit, each CR has a two-unit cache capacity, and each link has a one-unit transmission cost. Figure 1 shows the communication and caching in RSCB. For RSCB, CR_5 and CR_6 receive two coded-from-original blocks generated by CR_3 and CR_4, respectively. Coded-from-original blocks ocb_1 and ocb_2 are received from CR_1, where

$$\begin{aligned}
ocb_1 &= \alpha_{11} ob_1 + \alpha_{12} ob_2,\\
ocb_2 &= \alpha_{21} ob_3 + \alpha_{22} ob_4.
\end{aligned} \tag{6}$$

The total transmission cost is eight units. Figure 2 shows the conventional ICN communication. CR_2 receives the four original blocks (ob_1, ob_2, ob_3, ob_4) from CR_3 and CR_4 and then forwards these original blocks to CR_1. CR_1 responds to Interest $I(p: \{2, 4\}, 2)$ with two original blocks, ob_2 and ob_4, and Interest $I(p, \{1, 3\}, 2)$ with two further original blocks, ob_1 and ob_3. The total transmission cost is 12 units. Therefore, our proposed solution saves 33% of the transmission cost compared with conventional ICN and 20% more than our previous work [23].

3.3. Caching in RSCB. In RSCB, the original/coded-from-original blocks are cached by CRs to respond to future Interests. In order to provide consumers with sufficient linearly independent blocks in a single round, none of coded blocks that were encoded by coded blocks are cached in the network. The coded-from-original block only can be cached by a single CR, which is the immediate downstream neighbor of the CR that generated the block. Thus, the two coded-from-original blocks, ocb_1 and ocb_2, will be cached only by CR_2 (Figure 1(b). The coded-from-original block ocb_1 can be used as the original block ob_1 or ob_2 to respond to future Interests. When cache replacement happens, CR encodes several original blocks into one coded-from-original block to release the caching space. This ensures that all information contained in the original blocks is retained in the CR.

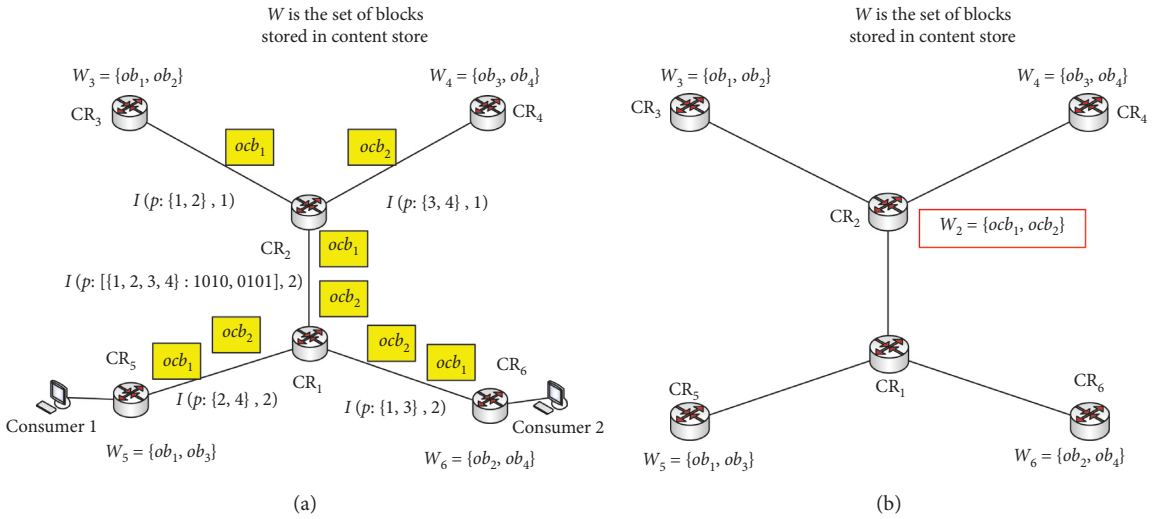

Figure 1: An example of RSCB. (a) Communication in RSCB. (b) Caching in RSCB.

Inputs

(1) Input Interest: $I(p, [S, b(S)], s)$

Steps

(1) Reconstruct subsets $[S_1, s_1]$,. . .and $[S_m, s_m]$ according to $b(S) = \{b(S_1), \ldots, b(S_m)\}$, where $s_i = |S_i|$, for $i = 1, \ldots, m$;
(2) **if** $S \neq \cup_i S_i$, $i = 1, \ldots, m$ **then**
(3) Determine subset $[S_{m+1}, s_{m+1}]$, where $S_{m+1} = S \backslash \cup_i S_i$, $s_{m+1} = 1$;
(4) **end if**
(5) **for** each subset $[S_i, s_i]$ **do**
(6) Use equation (2) to divide subset $[S_i, s_i]$ into several sub-subsets $[T_{i1}, t_{i1}], \ldots, [T_{ik}, t_{ik}]$ based on the forwarding interface of each chunk name (each and every sub-subset corresponds to an interface), where $t_{ij} = \min\{|T_{ij}|, s_i\}$, for $j = 1, \ldots, k$;
(7) **end for**
(8) Use equation (3) to generate new Interests by aggregating the sub-subsets $[T_1, t_1], \ldots, [T_n, t_n]$ according to the forwarding interfaces.

Algorithm 1: Separation.

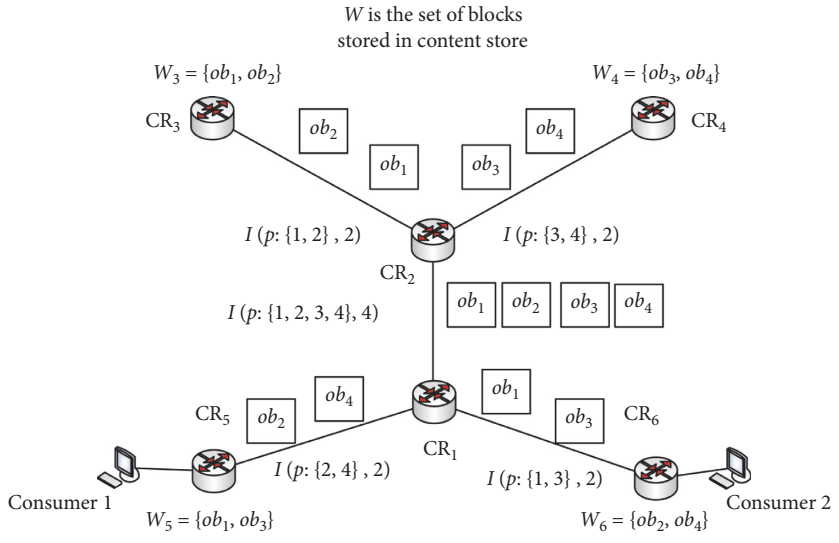

Figure 2: Communication in ICN.

4. Architecture

CCN architecture is the most popular architecture of ICN, and we have selected this architecture to implement RSCB. To enable network coding in CCN, some changes are required.

4.1. Content Publishing and Requesting. In CCN, content is split into several smaller-sized chunks, with each chunk identified by a unique hierarchical name. In RSCB, content is firstly divided into h generations and each generation is then divided into N chunks, i.e., *original blocks*. We denote content as $C_p = \{\{c_{p,1,1}, \ldots, c_{p,1,N}\}, \ldots, \{c_{p,h,1}, \ldots, c_{p,h,N}\}\}$, where p is the name of the content, N is a design parameter, and h can be calculated based on the content size and N. The content name p should contain information on h and N. For instance, the name of chunk $c_{p,2,1}$ is */sysu.edu.cn/largefile/h/N/2/1*, where */sysu.edu.cn/largefile/h/N* is the content name, p, 2 is the generation ID, and 1 is the chunk ID (referred to as the original block index).

A consumer will generate a set of Interests, $\{I_{p,1}, \ldots, I_{p,h}\}$, for each generation in order to request content C_p. Each Interest $I_{p,i}$ requests a set of original blocks, $\{c_{p,i,1}, \ldots, c_{p,i,N}\}$, where i is the number of generations. The number of Interests sent by the consumer depends on the flow control schemes which have been placed in the network. The consumer can send Interests either sequentially or simultaneously.

Since the forwarding paths of requests for different chunks generated by the same consumer will form a multicast tree, these requests will not meet in any intermediate CR on the multicast tree. Interests are responded to with chunks or coded blocks which are linear combinations of chunks that have been specified by the Interest. Random linear network coding (RLNC) is used to generate the coded blocks within each generation. For convenience, in the remainder of this paper, we will not explicitly state which generation each chunk belongs to. Our model makes the assumption that a chunk-based routing and flow control scheme is in place.

4.2. Interest and Data. All communications are driven by consumers in CCN. Consumers can receive chunks of content from multiple sources, which may include the content provider and CRs. A consumer interested in C_p will send a set of requests $I_p = \{i_{p,1}, \ldots, i_{p,N}\}$ with one request for each chunk. Before these requests are forwarded, the CR determines the forwarding interface of each chunk using the forwarding information base (FIB). Requests that have the same forwarding interface will be aggregated into a single Interest. Each Interest can contain multiple requests for a set of chunks, S_i, where S_i is the set of chunk names.

There are two types of CCN packets, Interests and data. In our model, the network coding information is appended to the selector field of the Interest packet and includes the set of chunk names S, the sub-Interests $b(S)$, and the number of required blocks n. The coefficient of the coded blocks, the caching flag Fq, is contained within the signed info field of the data packet. The data field of the data packet contains the original/coded block(s).

The Interest and data packets used in our model are formulated using the following method:

(i) $I(p, [S, B], m)$ defines an Interest for content C_p, where p is the content name, $S = (S_1 \cup S_2 \cup \ldots \cup S_n)$ is the set of chunk names, and $m = \max\{|S_1|, |S_2|, \ldots, |S_n|\} \leq |S|$ is the number of required blocks. $B = \{b(S_1), b(S_2), \ldots, b(S_n)\}$ is used to represent the subsets $S_1, S_2, \ldots, S_n$ and $S_i \not\subseteq S_j$, if $i \neq j$. For convenience, we express the Interest as $I(p, S, m)$.

(ii) $D(p, S, block, Fq)$ defines a data packet for content C_p containing *block*, which is either a chunk or a coded block (a linear combination of several chunks specified by set S). Fq is a caching flag which indicates whether the *block* is cacheable or not. If the *block* is a coded block, we obtain $S_D \supseteq S_I$, where S_D is the set of chunk names carried by the data and S_I is carried by the Interest.

4.3. Forwarding Module. The forwarding module of RSCB contains three components: the ContentStore (CS), the pending interest table (PIT), and the forwarding information base (FIB). The blocks received by the CRs are cached by the CS module. A CS entry can be formulated by $\mathrm{CS}(p, W, w)$, which is defined as follows:

(i) p: content name.

(ii) Index (W, w): W is a set of chunk names and w is the number of cached blocks. Since both the original and the coded-from-original blocks can be cached, we obtain $|W| \geq w$.

(iii) Data: the original or coded-from-original blocks.

In contrast with CCN, each CR interface in RSCB maintains a PIT. The PIT can have two types of entry, PIT-OUT and PIT-IN, which record information on Interests already sent or received to the interface, respectively. PIT-OUT and PIT-IN can be defined as follows:

(i) $\mathrm{PIT}_{\mathrm{out}-i}([p, S, s], facelist)$: this is a PIT-OUT entry that indicates that (p, S, s) is an aggregated Interest generated by aggregating several Interests received from interface(s) of $facelist$. The aggregated Interest has already been sent out over the interface i but a response has not been received from the upstream CR.

(ii) $\mathrm{PIT}_{\mathrm{in}-i}(p, S, s)$: this is a PIT-IN entry that indicates that (p, S, s) is an aggregated Interest generated by aggregating several Interests received from interface i. A response has not yet been sent over interface i.

The FIB is the same as for CCN. When an Interest is received by a CR, its CS is first consulted, followed by PIT and finally FIB. Data packets will be sent back to consumers using the same path that was created by the Interest, but in the opposite direction.

5. Communication Scheme

5.1. Forwarding Interest. When a CR receives an Interest $I(p, X, x)$ over interface f, the first step is to check its CS. If the CS contains all chunks or the coded-from-original blocks containing all the information in the Interested set X, the CR will respond to Interest $I(p, X, x)$ directly, as described in Algorithm 2. The complexity of Algorithm 2 is $O(n)$, where n is the number of coded blocks. If $|X| = x$, the CR responds to Interest $I(p, X, x)$ with the x cached blocks (chunks or coded-from-original blocks) without coding; otherwise, the CR will respond with the x coded blocks generated from the blocks cached in the CS, which contain the chunk information specified by set X. The caching flag, Fq, will be turned on, i.e., $Fq = 1$ if the block used to respond to the Interest is an original block or a coded-from-original block encoded by that CR; otherwise, $Fq = 0$. In RSCB, the CR performs network coding only if it will save on the transmission costs. For instance, as shown in Figure 1(a), CR responds to Interest $I(p, [\{1, 2, 3, 4\}: 1010, 0101], 2)$ with two coded-from-original blocks, ocb_1 and ocb_2, which were received from CR_3 and CR_4, respectively, without further coding. In this case, there is a saving on the cost of coding.

If the Interest cannot be satisfied by the CR, PIT-IN of the arrival interface f will be checked. If there is a matched entry $\mathrm{PIT}_{\mathrm{in}-f}(p, Z, z)$, CR will aggregate the PIT-IN entry and the Interest using equation (3) and will then update the PIT-IN entry $\mathrm{PIT}_{\mathrm{in}-f}(p, Z\prime, z\prime)$, where $Z\prime = X \cup Z$, $z\prime = \max\{x, z\}$. Therefore, the incremental Interest $(p, \Delta X, \Delta x) = (p, X, x) \backslash (p, W, w)$ will be determined.

The CR will split the incremental requests for $(p, \Delta X, \Delta x)$ into several Interests using Algorithm 1. If one of the Interests, e.g., (p, X_j, x_j), needs to be transmitted over the interface j, PIT-OUT of interface j will be obtained. If there is a matching PIT-OUT entry $\mathrm{PIT}_{\mathrm{out}-j}(p, V, v)$, a new incremental Interest for $(p, \Delta X_j, \Delta x_j) = (p, X_j, x_j) \backslash (p, V, v)$ will be generated using equation (5) and transmitted over interface j if $\Delta X_j \neq \mathrm{null}$. The PIT-OUT entry will then be updated to $\mathrm{PIT}_{\mathrm{out}-j}(p, V\prime, v\prime, facelist)$, where $V\prime = V \cup X_j$ and $v\prime = \max\{v, x_j\}$, and interface f will be added to $facelist$. Algorithm 3 describes the process used to forward an Interest. The complexity of Algorithm 3 is $O(n)$, where n is the number of sub-Interests.

5.2. Forwarding Data. When data packet $D(p, Y, block, Fq)$ is received by a CR over interface f, the PIT-OUT of interface f will be checked in the tables. If there is no PIT-OUT match, the data $D(p, Y, block, Fq)$ will be discarded directly since the CR has not requested the block; otherwise, the caching flag will be checked and the matching PIT-OUT entry will be updated according to Algorithm 4. If $Fq = 1$, the block carried by data will be cached into CS; otherwise, it will be temporarily cached into CACHE. The CR will then check whether more chunks can be obtained by decoding the blocks cached in CS and CACHE. If the CR has already received enough blocks of content p to recover the content, the chunks decoded from the received blocks will be cached into CS and all of the blocks of content p that were cached in CACHE will be deleted. In this case, CR can satisfy any Interest of content p.

If interface i is included in the $facelist$ of the matching PIT-OUT entry, the corresponding PIT-IN entry is $\mathrm{PIT}_{\mathrm{in}-i}(p, Z, z)$. If $Y \cap Z \neq \emptyset$, CR checks whether the $\mathrm{PIT}_{\mathrm{in}-i}(p, Z, z)$ can be satisfied using blocks cached in CS and CACHE. If it can be satisfied, CR will generate z linearly independent combinations of the blocks specified by the set Z and will send z data packets over interface i. Each data packet carries a coded block and $\mathrm{PIT}_{\mathrm{in}-i}(p, Z, z)$ is then deleted. Once the PIT-IN entries of all interfaces in $facelist$ of the matched PIT-OUT entry are satisfied, the coded blocks of content p that are cached in CACHE are deleted, as described by Algorithm 4. The complexity of Algorithm 4 is $O(nm)$, since the complexity of Algorithm 2 is $O(m)$, where n is the number of interfaces in the $facelist$ of the matched PIT-OUT and m is the number of coded blocks.

The network will try to deliver chunks without introducing extra traffic in order to increase the independence of the blocks cached in CRs. When the CR receives a data packet $D(p, Y, block, Fq)$ carrying a chunk (i.e., $|Y| = 1$), if $z = |Z|$, the data packet $D(p, Y, block, Fq)$ will be sent out over interface i without further processing or waiting for other data packets. The CR will then update $\mathrm{PIT}_{\mathrm{in}-i}(p, Z, z)$ to $\mathrm{PIT}_{\mathrm{in}-i}(p, Z\prime, z\prime)$, where $Z\prime = Z \backslash Y$, $z\prime = z - 1$. If $z = 0$, the CR will delete the PIT-IN entry. In this case, the time to download chunk Y will be reduced and the cost of coding and decoding is saved without introducing additional traffic.

5.3. Cache Policy. Network coding-enabled ICN (NC-ICN) will divide the content into n original blocks. For traditional NC-ICN, each coded block is a linear combination of the n original blocks. The n coded blocks will be cached by n' $(n' \geq n)$ CRs along their forwarding paths to a group of consumers. Figure 3(a) shows that for traditional NC-ICN, network $N1$ will provide m coded blocks, $cb_1, \ldots, cb_m$, to consumers in group $G1$ and network $N2$ will provide the remaining $(n - m)$ coded blocks. During the process of responding to consumers in group $G1$, $m\prime$ $(m\prime \geq m)$ coded blocks generated by $cb_1, \ldots, cb_m$ will be cached by multiple CRs in network $N1$, while n' $(n' \geq (n - m))$ coded blocks will be cached by multiple CRs in network $N2$. At a later time, when the consumers in $G2$ multicast their Interests for n coded blocks of content p, these Interests will be received first by CRs in network $N1$. Each CR will respond to the Interest with its cached coded blocks independently, and thus t $(t > m)$ coded blocks cached in network $N1$ may be provided to consumers in $G2$. However, the maximum number of independent coded blocks that a consumer can receive from network $N1$ is m. In this case, at least $(t - m)$ blocks are not beneficial to the consumer and are a waste of resources. Therefore, the conventional in-network caching strategy is not suitable for NC-ICN.

To address this issue, we propose to introduce a simple cache mechanism to guarantee that the blocks provided to consumers will be independent. In RSCB, only original blocks and coded-from-original blocks will be cached by CRs. None of coded blocks that were encoded by other

```
Input: I(p, X, x) ← Interest arriving on interface f,
CS_p(W, w)←W is the set of blocks specified by I(p, X, x) and w is the number of blocks;
(1) if |X| = x or x = w then
(2) for each block cs_i in CS_p do
(3) if cs_i is an original block then
(4) Fq = 1;
(5) else
(6) Fq = 0;
(7) end if
(8) Create a data packet D(p, Y_i, cs_i, Fq) and send over interface f;
(9) end for
(10) else
(11) Generate x coded blocks, ob_1, ..., and ob_x, using the blocks in CS_p;
(12) if all the blocks in CS_p are original blocks then
(13) Fq = 1;
(14) else
(15) Fq = 0;
(16) end if
(17) for each coded block cb_i do
(18) Generate a data packet D(p, Y_i, cb_i, Fq) and send over interface f;
(19) end for
(20) end if
```

Algorithm 2: Responding Interest.

```
Input: Interest I(p, X, x); interface f
(1) if CS matches then
(2) Respond to Interest according to Algorithm 2;
(3) else
(4) if PIT-IN of interface f matches then
(5) Update PIT-IN;
(6) else
(7) Establish a new PIT-IN entry;
(8) end if
(9) Calculate the incremental Interest ΔI;
(10) if ΔI ≠ null then
(11) Separate the incremental Interest ΔI into several sub-Interests according to Algorithm 1;
(12) for each sub-Interest I_i which needs to be sent over interface i do
(13) if PIT-OUT of i matches then
(14) Update PIT-OUT;
(15) Calculate the incremental Interest ΔI_i;
(16) if ΔI_i = null then
(17) return;
(18) end if
(19) else
(20) Establish a new PIT-OUT entry;
(21) end if
(22) Send Interest I_i (or ΔI_i) from interface i;
(23) end for
(24) end if
(25) end if
```

Algorithm 3: Forwarding Interest.

coded blocks can be cached in the network. The received/decoded original blocks can be cached by any CR. The coded-from-original blocks can only be cached by a single CR, which is the immediate downstream neighbor of the CR that generated the block. Thus, any n coded-from-original blocks cached in the network will be linearly independent, where n is the number of blocks required to recover the content. Fq in data packet $D(p, S, block, Fq)$ is a caching flag that indicates whether the *block* is cacheable or not.

Input: data packet $D(p, Y, block, Fq)$; interface f
(1) **if** PIT-OUT of interface f matches **then**
(2) **if** $Fq = 1$ **then**
(3) Cache data into CS;
(4) **else**
(5) Cache data into CACHE;
(6) **end if**
(7) **for** each interface i in the $facelist$ of the matched PIT-OUT **do**
(8) Find the matched PIT-IN entry, i.e. $\mathrm{PIT}_{\mathrm{in}-i}(p, Z, z)$, where $Y \cap Z \neq \varnothing$;
(9) **if** the matched PIT-IN entry of interface i is satisfied **then**
(10) Respond to the PIT-IN entry $\mathrm{PIT}_{\mathrm{in}-i}(p, Z, z)$, according to Algorithm 2;
(11) Delete the PIT-IN entry;
(12) **else**
(13) **if** the block carried by data is an original block and $|Z| = z$ **then**
(14) Send data $D(p, Y, block, Fq)$ over interface i;
(15) Update $\mathrm{PIT}_{\mathrm{in}-i}(p, Z, z)$ to $\mathrm{PIT}_{\mathrm{in}-i}(p, Z\prime, z\prime)$, where $Z\prime = Z \backslash Y$, $z\prime = z - 1$;
(16) **end if**
(17) **end if**
(18) **end for**
(19) **if** the matched PIT-OUT of interface f is satisfied **then**
(20) Delete the PIT-OUT entry;
(21) **end if**
(22) **else**
(23) Discard data $D(p, Y, block, Fq)$;
(24) **end if**

Algorithm 4: Forwarding data.

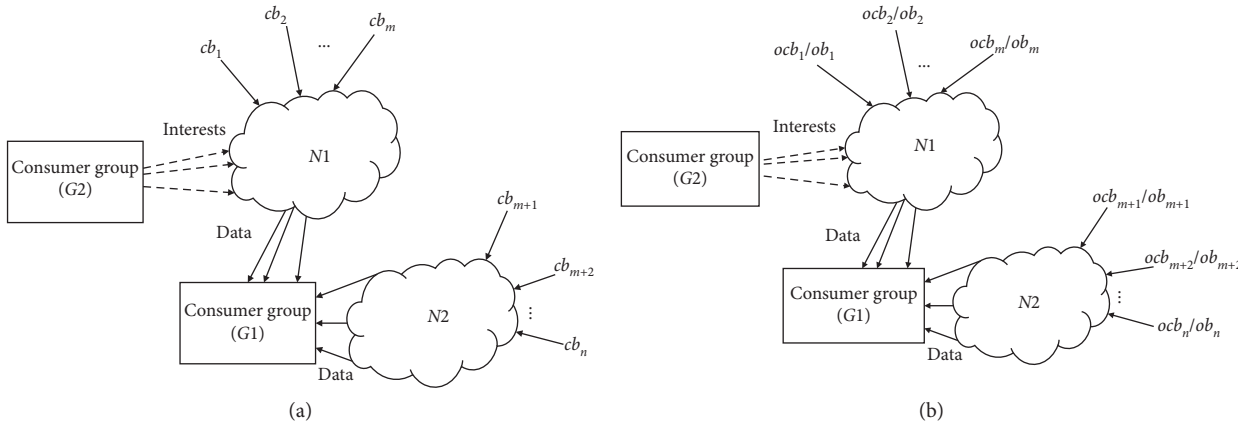

Figure 3: Caching. (a) Caching in ICN. (b) Caching in RSCB.

Since CRs have limited storage capacity, a cache replacement policy is required. When cache replacement occurs, the candidate content p that is to be discarded is selected using the existing content-level cache replacement policy, e.g., least recently used (LRU). Assume t units of cache space are required to cache newly received/decoded blocks and the cache space used to cache the candidate content p is n units (one unit for each block). If $t \geq n$, content p is deleted and $t = t - n$. The first step is repeated until only some of cached blocks of the candidate content p_i need to be discarded to cache new blocks, i.e., $t < n_i$. Chunk-level cache replacement is then introduced to discard blocks in content p_i. Firstly, any original blocks that are contained in the cached coded-from-original blocks are discarded, i.e., the information contained in the original blocks ob_1 and ob_2 may also be contained in the coded-from-original blocks $ocb_1 = \alpha_{11} ob_1 + \alpha_{12} ob_2$. Secondly, the remaining original blocks are coded into a single coded-from-original block. Finally, the received coded-from-original blocks are randomly discarded. These three steps are performed in turn until there is sufficient space for the newly received/decoded blocks, as described in Algorithm 5. The complexity of Algorithm 5 is $O(n)$, where n is the number of evicted content. If content is rarely accessed, only the coded-from-original blocks will be cached to respond to future Interests. Since a single coded-from-original block can respond to an aggregated Interest containing multiple requests for different chunks sent by multiple consumers, our chunk-level

Input: t newly received/decoded blocks
(1) Determine the candidate content p with n blocks using LRU;
(2) **if** $t \geq n$ **then**
(3) Let $t = t - n$;
(4) Discard content p;
(5) **if** $t > 0$ **then**
(6) go to step 1;
(7) **end if**
(8) **else**
(9) Discard the n_1 original blocks which are contained in the coded-from-original blocks;
(10) Let $\Delta t_1 = t - n_1$;
(11) **if** $\Delta t_1 > 0$ **then**
(12) Encode the remaining n_2 original blocks into a single coded blocks;
(13) Let $\Delta t_2 = \Delta t_1 - n_2 + 1$;
(14) **if** $\Delta t_2 > 0$ **then**
(15) Randomly discard Δt_2 received coded-from-original blocks;
(16) **end if**
(17) **end if**
(18) **end if**
(19) Cache the newly received/decoded blocks into the CS;

Algorithm 5: Cache replacement strategy.

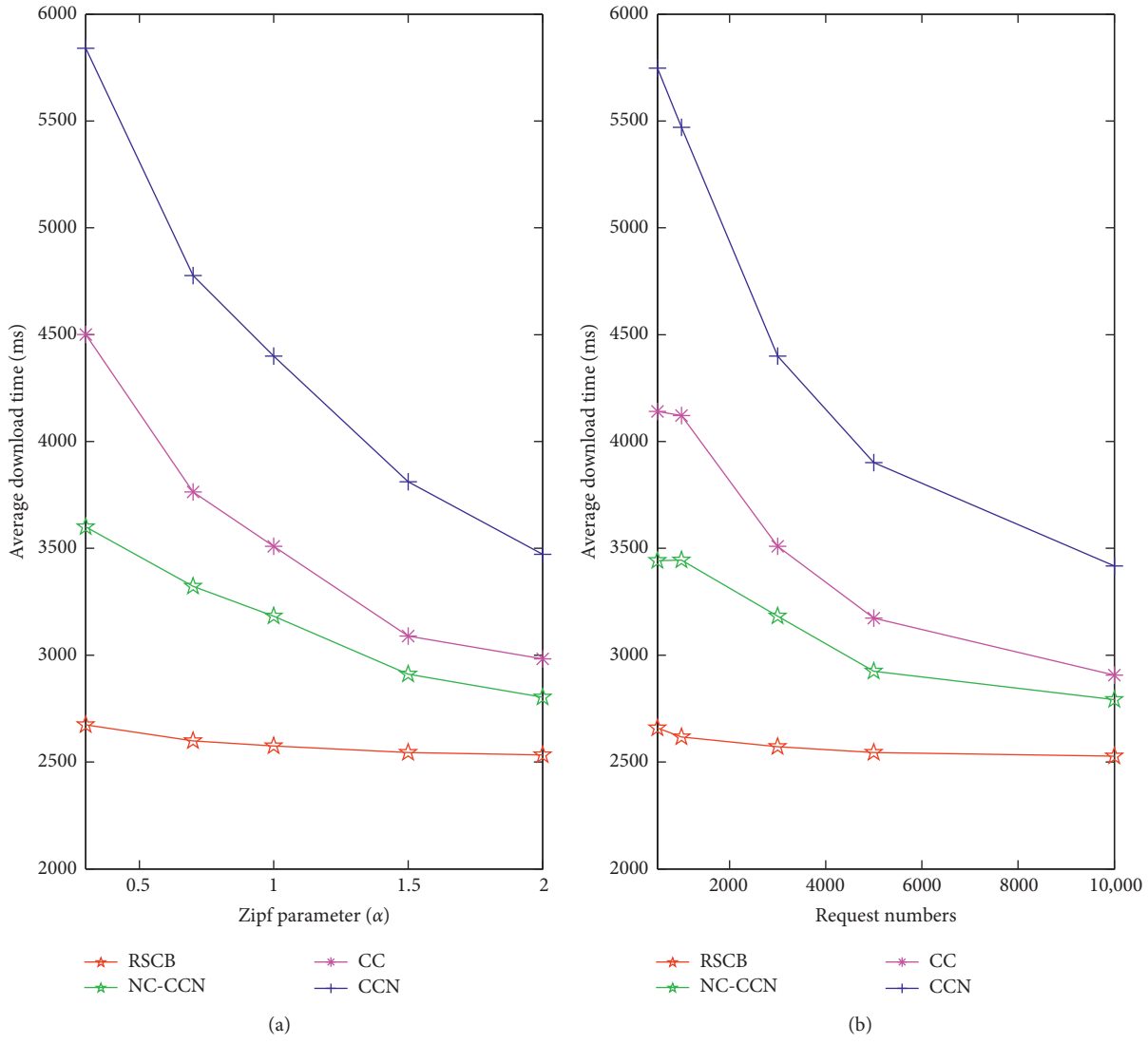

Figure 4: Average download time. (a) Zipf parameter VS average download time. (b) Request numbers VS average download time.

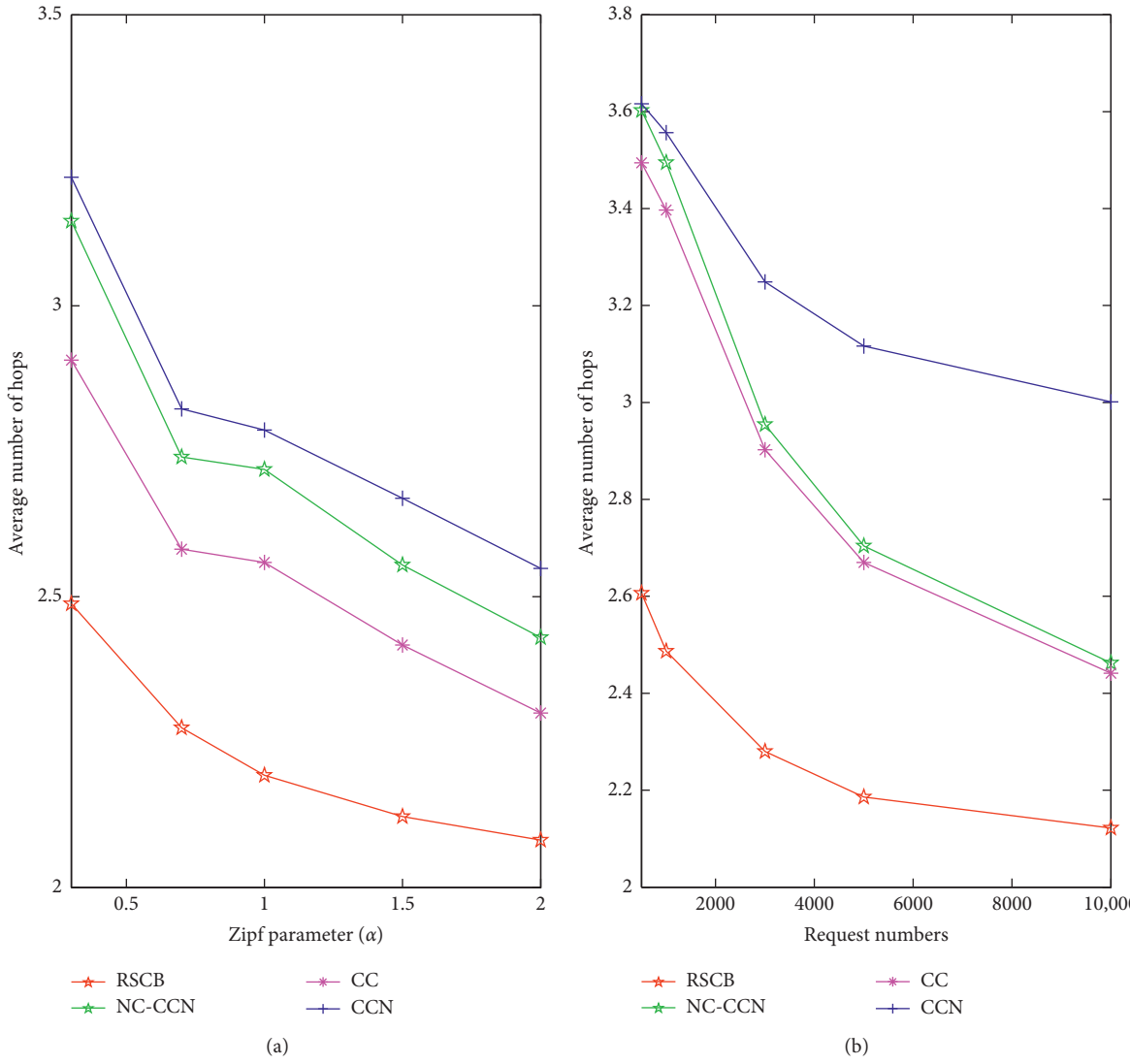

Figure 5: Average number of hops. (a) Zipf parameter VS average number of hops. (b) Request numbers VS average number of hops.

cache replacement policy can effectively increase the cache efficiency.

6. Simulation

In this section, the performance of our model is investigated by comparison with other three schemes: chunk-level CCN strategy (CCN) [4], NC-CCN [21], and CodingCache (CC) [16]. The caching strategy leave copy everywhere (LCE) is incorporated into the above strategies. In LCE, each block or chunk is cached by all CRs on the forwarding path between the content provider and the consumer.

6.1. Simulation Model. BRITE [27, 28] was used to generate the network topology, since it can roughly reflect the actual Internet topology. The Dijkstra algorithm was used to generate the FIB tables. All links have a bandwidth of 1 Gbps. There were a total of 1000 end hosts that were connected to 100 CRs and 10 original content providers were randomly connected to the CRs. 10,000 files were equally partitioned into 400 classes. Each content packet was 1 GB and was divided into 10 generations with each generation containing 10 chunks; each chunk size was 10 MB. In our simulation, only chunks in the same generation could be encoded. The content popularity follows a Zipf distribution with $\alpha \in [0.3, 0.7, 1, 1.5, 2]$. Interests sent by consumers follow a Poisson process. The request number was defined as the number of Interests sent by consumers during the processing period. In our simulations, each CR was equally configured to have a cache space of 0.1%, 0.25%, 0.5%, 1%, and 2% of the overall content catalog size. The default cache size of each CR was set to 10 GB for caching, i.e., 1% of the total content catalog size. Random linear network coding (RLNC) was used for coding. The size of a finite field was 2^8 [29]. The coefficient vector and the generation ID are contained within the signed info field of the data packet. The performance of all four strategies was evaluated under the same simulation environment.

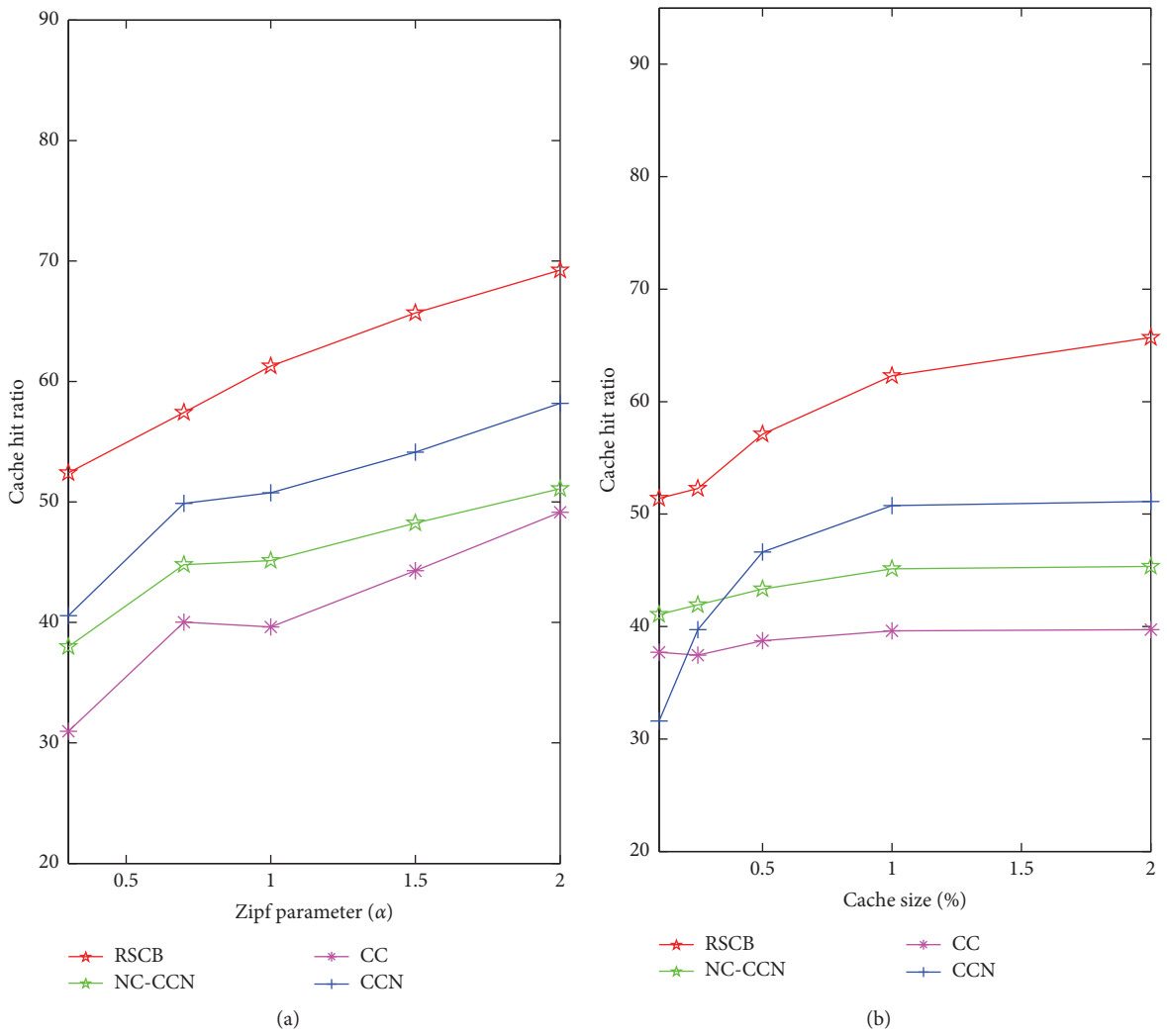

Figure 6: Cache hit ratio. (a) Zipf parameter VS cache hit ratio. (b) Cache size VS cache hit ratio.

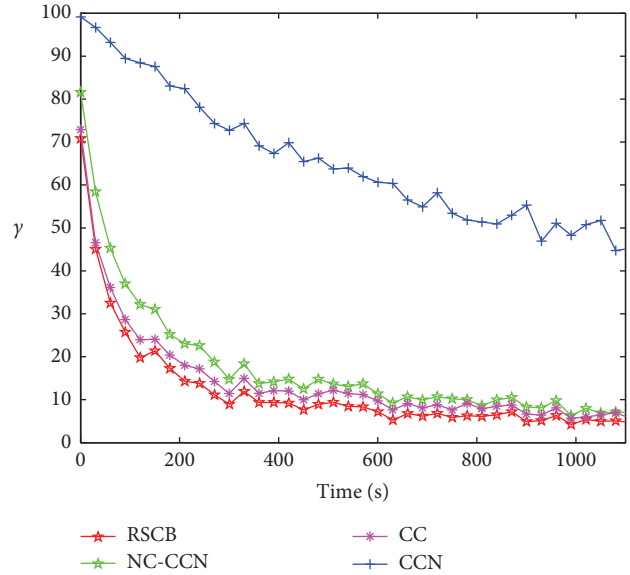

Figure 7: Server hit reduction ratio.

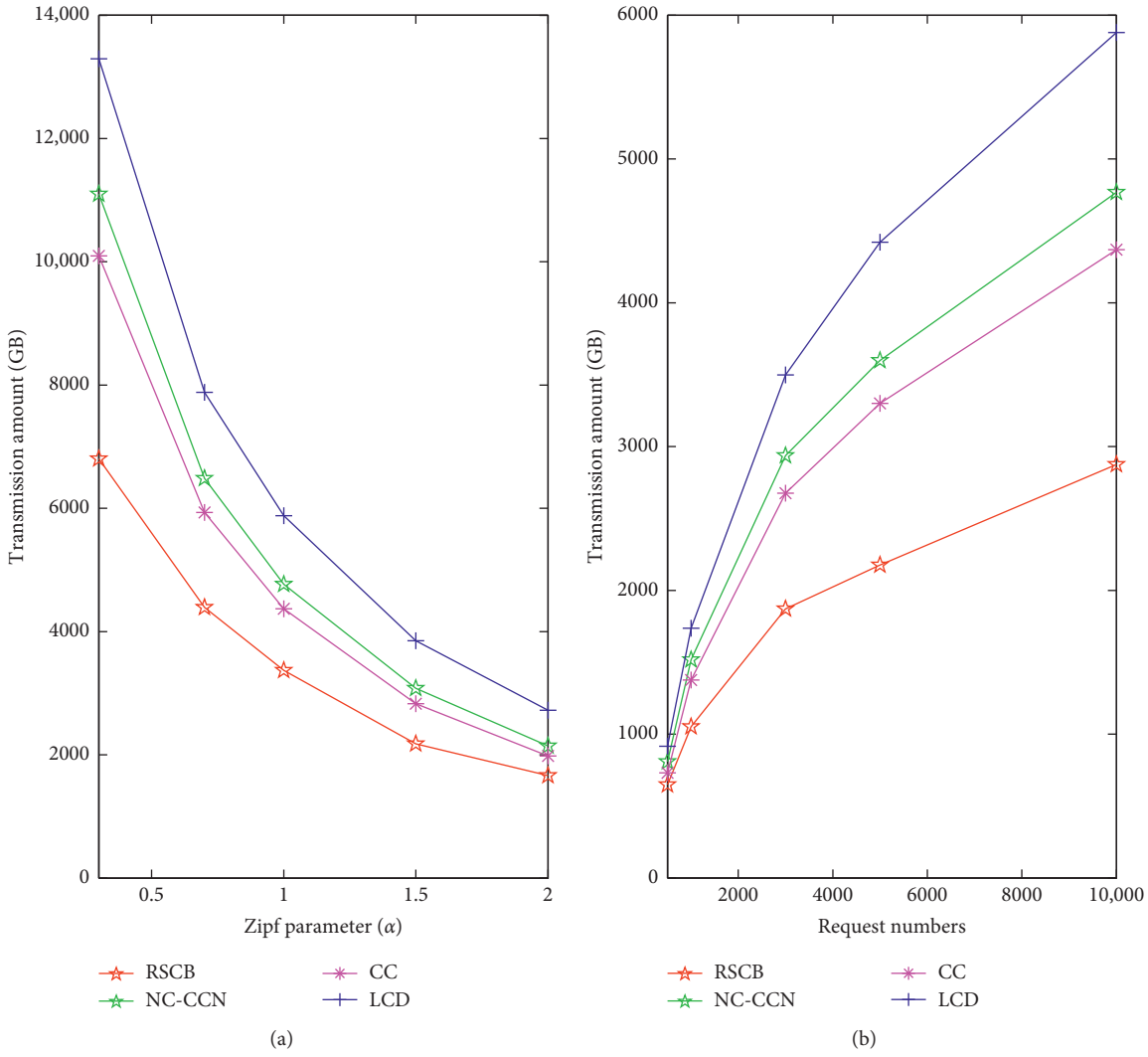

Figure 8: Traffic. (a) Zipf parameter VS transmission amount. (b) Request numbers VS transmission amount.

The following parameters were used for the evaluation:

(i) Average download time: the average time for consumers to download each successfully received content request response.

(ii) Average number of hops: the average number of hops for each successfully received chunk from the provider to the consumer.

(iii) Cache hit ratio: the ratio of the number of Interests that were satisfied by the caches to the number of Interests that were satisfied by either the caches or the server.

(iv) Server hit reduction ratio $\gamma(t)$:

$$\gamma(t) = \frac{\sum_{i=1}^{N(t)} w_i(t)}{N(t)}, \tag{7}$$

where $w_i(t) = 0$ if the chunk i is sent from a cache or an aggregated Interest; otherwise, $w_i(t) = 1$. $N(t)$ is the number of chunks received by all consumers. t indicates that the data were collected from time $(t - \Delta t)$ to t [20].

(v) Traffic: the total traffic to deliver the data packets over the whole request process.

(vi) Average number of Interests: the average number of Interest packets that were handled by each CR for each chunk that was successfully received by the consumer, as in [21].

6.2. Simulation Results. Due to its network coding-based content delivery and caching strategies, our proposed RSCB always achieves the best performance in terms of having the shortest average download time, the highest cache hit ratio, the lowest server hit reduction ratio, and the lowest transmission volume. RSCB ensures that consumers will receive sufficient independent coded blocks within a single round and the coded blocks cached by the CRs can be used as multiple chunks.

Figure 4 plots the average download time of the four caching strategies for different system parameters. The

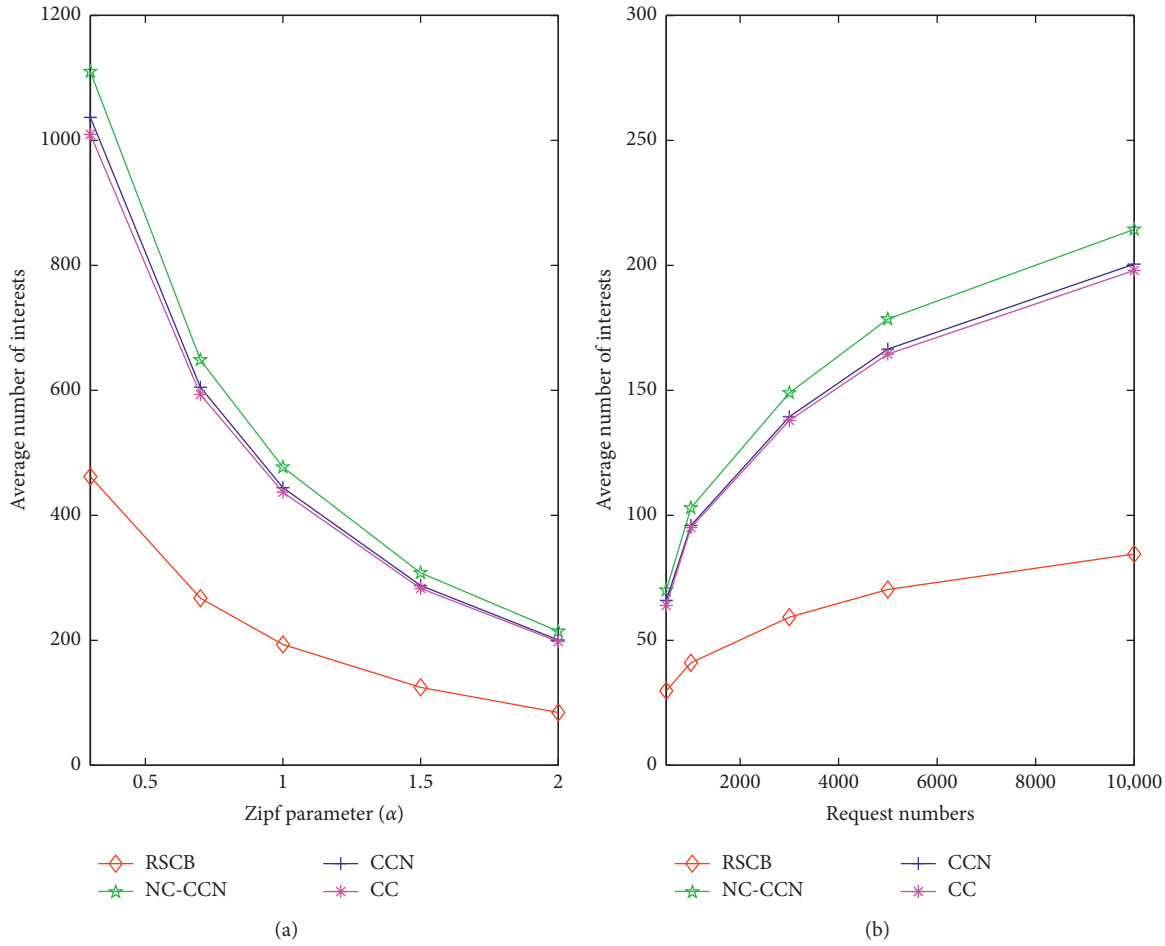

Figure 9: Average number of Interests. (a) Zipf parameter VS average number of Interests. (b) Request numbers VS average number of Interests.

average download time decreases as the Zipf parameter α is increased, as shown in Figure 4(a), since a larger Zipf parameter α indicates that the Interests sent by consumers are concentrated on a smaller set of contents. As the number of requests increases, chunks that have already been requested will be cached on more CRs and thus consumers can retrieve chunks directly from the CRs, which are much closer to the consumers. Therefore, the average download time will be reduced (Figure 4(b)). RSCB performs much better even for a small Zipf parameter and a low number of Interests, since it can retrieve chunks or coded blocks from multiple CRs simultaneously. Compared with other schemes, RSCB provides consumers with enough independent coded blocks in a single round.

In RSCB, one coded-from-original block can be used to respond to an aggregated Interest for multiple chunks requested by different consumers. For instance, the coded-from-original block, $ocb_1 = \alpha_{11}ob_1 + \alpha_{12}ob_2$, can satisfy the Interest for chunk ob_1 from consumer 1 and the Interest for chunk ob_2 from consumer 2, as shown in Figure 1(b). Thus, RSCB achieves the best caching performance, in terms of average download hops (Figure 5), cache hit ratio (Figure 6), and server hit reduction ratio (Figure 7).

Figure 8 shows the traffic for different caching schemes, and it can be seen that RSCB has the lowest transmission volume. Moreover, we can see that as the number of requests increases, RSCB has a higher traffic saving too due to its Interest aggregation scheme which saves on traffic required to deliver $n - (n_1 + n_2)$ blocks, as per equation (3).

RSCB can also aggregate Interests for different chunks into a single Interest. As shown in Figure 9, the average number of Interests processed by the CR is much lower compared with other schemes. In ICN, a consumer requests content with N chunks by sending out N Interests, and thus the CR needs to process N Interests. However, in RSCB, only one aggregated Interest containing several Interests will be processed by the CR. This can reduce the cost of transmitting and processing the Interest.

7. Conclusion and Discussion

In this paper, we have proposed a request-specific coded-block strategy to reduce the transmission volume. Additionally, a chunk-level on-path non-cooperative coded caching and replacing strategy has been proposed to

improve the caching efficiency. Our method enables a consumer to multicast a set of Interests in order to obtain multiple content chunks simultaneously from multiple CRs. The traffic can be reduced by encoding chunks that meet in an intermediate CR and have been requested by different consumers. A novel Interest forwarding-responding strategy has been proposed to guarantee that the minimum number of coded blocks will be requested and that the blocks will be linearly independent. A network coding-based caching and replacing mechanism has been designed to guarantee that the cached blocks can be reused. A chunk-level coded cache replacement strategy has been proposed to discard blocks. Rather than discarding the original blocks, the CR will encode the original blocks into a single coded-from-original block to release cache space when cache replacement is required. A single coded-from-original block can satisfy multiple Interests from different consumers for a set of its component original blocks. Therefore, this will increase the caching diversity without requiring extra cache space. The simulation results have confirmed that the RSCB scheme outperforms the other three strategies.

However, although there are many benefits in deploying network coding in ICN, it also introduces some additional costs for computation and communication. Some studies have already proven that RLNC is a practical method which has acceptable costs. Since ICN is a new architecture, there are still many issues that need to be resolved before ICN can be deployed, such as efficient operation of PIT and FIB at a chunk level [30, 31].

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (nos. 61571141, 61702120, 61972104, and 61902080), the National Key Research and Development Project (no. SQ2019YFB180098), the Guangdong Natural Science Foundation (no. 2017A030310591), the Guangdong Provincial Application-Oriented Technical Research and Development Special Fund Project (nos. 2017B010125003 and 2015B010131017), the Key Areas of Guangdong Province (nos. 2019B010118001 and 2017B030306015), the Guangdong Future Network Engineering Technology Research Center (no. 2016GCZX006), the Science and Technology Program of Guangzhou (no. 201604016108), the Project of Youth Innovation Talent of Universities in Guangdong (nos. 2017KQNCX120 and 2016KQNCX091), the Guangdong Science and Technology Development Project (no. 2017A090905023), the Key Projects of Guangdong Science and Technology, and the Science and Technology Project in Guangzhou (no. 201803010081).

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] I. U. Din, S. Hassan, M. K. Khan, M. Guizani, O. Ghazali, and A. Habbal, "Caching in information-centric networking: strategies, challenges, and future research directions," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 2, pp. 1443–1474, 2018.

[3] S. Lee, I. Yeom, and D. Kim, "T-caching: enhancing feasibility of in-network caching in icn," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 7, pp. 1486–1498, 2020.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," *Communications of the ACM*, vol. 55, no. 1, pp. 117–124, 2012.

[5] R. Ahlswede, N. Ning Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[6] B. Saleh and D. Qiu, "Performance analysis of network-coding-based p2p live streaming systems," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2140–2153, 2016.

[7] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 4, pp. 2235–2245, Miami, FL, USA, March 2005.

[8] M. Karmoose, M. Cardone, and C. Fragouli, "Simplifying wireless social caching via network coding," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5512–5525, Nov 2018.

[9] C. Xu, P. Wang, C. Xiong, X. Wei, and G.-M. Muntean, "Pipeline network coding-based multipath data transfer in heterogeneous wireless networks," *IEEE Transactions on Broadcasting*, vol. 63, no. 2, pp. 376–390, 2017.

[10] M. Bilal and S.-G. Kang, "Network-coding approach for information-centric networking," *IEEE Systems Journal*, vol. 13, no. 2, pp. 1376–1385, 2019.

[11] H. R. Sadjadpour, "A new design for information centric networks," in *Proceedings of the 48th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, Princeton, NJ, USA, March 2014.

[12] J. Wang, J. Ren, K. Lu, J. Wang, S. Liu, and C. Westphal, "An optimal cache management framework for information-centric networks with network coding," in *Proceedings of the 2014 IFIP Networking Conference*, pp. 1–9, Trondheim, Norway, June 2014.

[13] Q. Xiang, H. Zhang, J. Wang, G. Xing, S. Lin, and X. Liu, "On optimal diversity in network-coding-based routing in wireless networks," in *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 765–773, Kowloon, Hong Kong, April 2015.

[14] J. Llorca, A. M. Tulino, K. Guan, and D. C. Kilper, "Network-coded caching-aided multicast for efficient content delivery," in *Proceedings of the 2013 IEEE International Conference on Communications (ICC)*, pp. 3557–3562, Budapest, Hungary, 2013.

[15] P. Talebifard, H. Nicanfar, and V. C. Leung, "A content centric approach to energy efficient data dissemination," in

Proceedings of the 2013 IEEE International Systems Conference (SysCon), pp. 873–877, Orlando, FL, USA, April 2013.

[16] Q. Wu, Z. Li, and G. Xie, "Codingcache: multipath-aware ccn cache with network coding," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking-ICN'13*, pp. 41-42, USA, 2013.

[17] M.-J. Montpetit, C. Westphal, and D. Trossen, "Network coding meets information-centric networking: an architectural case for information dispersion through native network coding," in *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications-NoM'12*, pp. 31–36, USA, June 2012.

[18] W.-X. Liu, S.-Z. Yu, G. Tan, and J. Cai, "Information-centric networking with built-in network coding to achieve multi-source transmission at network-layer," *Computer Networks*, vol. 115, pp. 110–128, 2017.

[19] J. Saltarin, E. Bourtsoulatze, N. Thomos, and T. Braun, "Netcodccn: a network coding approach for content-centric networks," in *Proceedings of the IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, San Francisco, CA, USA, April 2016.

[20] D. Nguyen, M. Fukushima, K. Sugiyama, and A. Tagami, "CoNAT: a network coding-based interest aggregation in content centric networks," in *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*, pp. 5715–5720, London, UK, June 2015.

[21] G. Zhang and Z. Xu, "Combing CCN with network coding: an architectural perspective," *Computer Networks*, vol. 94, pp. 219–230, 2016.

[22] C. Shan, J. Cai, Y. Liu, and J. Luo, "Node importance to community based caching strategy for information centric networking," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 21, Article ID e4797, 2019.

[23] Y. Liu and S.-Z. Yu, "Network coding-based multisource content delivery in content centric networking," *Journal of Network and Computer Applications*, vol. 64, pp. 167–175, 2016.

[24] J. Wang, J. Ren, K. Lu, J. Wang, S. Liu, and C. Westphal, "A minimum cost cache management framework for information-centric networks with network coding," *Computer Networks*, vol. 110, pp. 1–17, 2016.

[25] N. Lal, S. Kumar, and V. K. Chaurasiya, "A network-coded caching-based multicasting scheme for information-centric networking (ICN)," *Iranian Journal of Science and Technology*, vol. 43, no. 3, pp. 427–438, 2019.

[26] J. Saltarin, T. Braun, E. Bourtsoulatze, and N. Thomos, "Popnetcod: a popularity-based caching policy for network coding enabled named data networking," 2019, http://arxiv.org/abs/1901.01187.

[27] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: an approach to universal topology generation," in *Proceedings of the Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 346–353, Cincinnati, OH, USA, August 2001.

[28] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in internet topologies," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 2, pp. 18–28, Apr. 2000.

[29] C. Fragouli, J. Widmer, and J.-Y. Le Boudec, "Efficient broadcasting using network coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 2, pp. 450–463, 2008.

[30] Y. Wang, K. He, H. Dai et al., "Scalable name lookup in ndn using effective name component encoding," in *Proceedings of the ICDCS '12*, pp. 688–697, June 2012.

[31] T. Song, H. Yuan, P. Crowley, and B. Zhang, "Scalable name-based packet forwarding: from millions to billions," in *Proceedings of the ACM-ICN'15*, pp. 19–28, San Francisco, CA, USA, 2015.