PAPER *Special Section on Parallel and Distributed Computing and Networking*

Color-Based Cooperative Cache and Its Routing Scheme for Telco-CDNs

Takuma NAKAJIMA[†a], ***Nonmember***, **Masato YOSHIMI**[†b], **Celimuge WU**[†c], ***Members***, ***and*** **Tsutomu YOSHINAGA**[†d], ***Senior Member***

SUMMARY Cooperative caching is a key technique to reduce rapid growing video-on-demand's traffic by aggregating multiple cache storages. Existing strategies periodically calculate a sub-optimal allocation of the content caches in the network. Although such technique could reduce the generated traffic between servers, it comes with the cost of a large computational overhead. This overhead will be the cause of preventing these caches from following the rapid change in the access pattern. In this paper, we propose a light-weight scheme for cooperative caching by grouping contents and servers with color tags. In our proposal, we associate servers and caches through a color tag, with the aim to increase the effective cache capacity by storing different contents among servers. In addition to the color tags, we propose a novel hybrid caching scheme that divides its storage area into colored LFU (Least Frequently Used) and no-color LRU (Least Recently Used) areas. The colored LFU area stores color-matching contents to increase cache hit rate and no-color LRU area follows rapid changes in access patterns by storing popular contents regardless of their tags. On the top of the proposed architecture, we also present a new routing algorithm that takes benefit of the color tags information to reduce the traffic by fetching cached contents from the nearest server. Evaluation results, using a backbone network topology, showed that our color-tag based caching scheme could achieve a performance close to the sub-optimal one obtained with a genetic algorithm calculation, with only a few seconds of computational overhead. Furthermore, the proposed hybrid caching could limit the degradation of hit rate from 13.9% in conventional non-colored LFU, to only 2.3%, which proves the capability of our scheme to follow rapid insertions of new popular contents. Finally, the color-based routing scheme could reduce the traffic by up to 31.9% when compared with the shortest-path routing.

key words: *cooperative caching, routing algorithm, color tags, hybrid caching, dynamic content popularity*

1. Introduction

Internet traffic is rapidly growing due to the wide spread of Video-on-Demand (VoD) services. It is estimated that the video traffic will reach more than 80% of the whole internet traffic in 2020 [1]. Since such enormous traffic will cause many congested links and degrade network performances, VoD service providers usually place their contents on Content Delivery Networks (CDNs) which are global-scale networks consisting of many cache servers. Although CDNs could reduce the video traffic, their servers are usually located outside of the Internet Service Provider (ISP) networks. Several CDN providers place their cache servers in ISP networks [2]–[4] and try to engineer the traffic to further reduce traffic [5]. However, such servers are only in limited locations, causing congested links near the cache servers. Moreover, ISPs cannot accept CDNs to engineer the traffic since CDN providers have no global knowledge of the underlying network [5]. This implicates that CDNs still cannot reduce traffic on peering links to the contents servers as well as inside of the ISP networks, introducing congested links as well as data traffic overhead. Although such problems could be relieved by introducing new network fabrics, such solution is expensive and not scalable since the needed network fabrics are proportional with the data growing. Hence, it is a major challenge for ISPs to reduce the traffic under reasonable cost overhead.

Several ISPs are planning to build their own CDNs by placing cache servers in their networks, which is called Telco-CDNs [5], [6]. They reduce the traffic on the peering links as well as internal communication links by confining the video requests to their networks. However, the obtained traffic reduction is not enough due to the limited storage capacity compared to the large library of contents. Moreover, an inefficient allocation of contents often leads to congested links because popular contents are responsible for most of the requests that concentrate traffic on a few servers and links.

Recent studies try to reduce the video traffic by increasing the effective cache capacity. They apply a cooperative caching strategy that aggregates multiple cache storages by sharing contents among cache servers [5], [7]. In Telco-CDNs, an ISP manages both the physical network and cache servers, which makes possible to realize the cooperative caching by traffic engineering. Existing schemes often periodically calculate a sub-optimal allocation of contents in the network by solving an optimization problem by heuristics approach such as Genetic Algorithm (GA). Although such strategies could eliminate a large amount of traffic, they often require hours of calculation overhead. Such long calculation will cause mismatches in cache allocations since access patterns change by 20–60% every hour [8]. Hybrid caching strategies are also proposed to follow the changes in access patterns [9], however, they do not support cooperative caching. In addition to the adopted cache aggregation scheme, the routing algorithm responsible for handling the data between caches plays an important role in the traffic re-

Manuscript received January 6, 2017.
Manuscript revised May 17, 2017.
Manuscript publicized July 14, 2017.

[†]The authors are with the University of Electro-Communications, Chofu-shi, 182–8585 Japan.
a) E-mail: tnakajima@comp.is.uec.ac.jp
b) E-mail: yoshimi@is.uec.ac.jp
c) E-mail: clmg@is.uec.ac.jp
d) E-mail: yosinaga@is.uec.ac.jp

DOI: 10.1587/transinf.2017PAP0014

duction. In fact, it is not a trivial task to find a location of a cached content that might change in time.

In this paper, we consider that important factors of traffic reduction are content distribution and duplication of popular contents. In fact, the content distribution increases the effective cache capacity by storing different contents among cache servers, while the duplication of popular contents improves cache hit rates of servers. From these facts, we propose in this work a new methodology that takes into consideration the content distribution and the popular content duplication, by grouping caches and servers using a novel color tag scheme. These color tags are efficiently distributed to the caches and servers through a light-weight color distribution scheme. A cooperative hybrid caching scheme is also proposed to follow rapid changes in access patterns that maintains hit rates when an access pattern changes rapidly. Moreover, an efficient routing scheme is proposed to further reduce traffic utilizing the color information provided by the color tag scheme. The proposed routing scheme transfers requests to the nearest server that matches the color with the desired content, while it requires only a small size of an additional routing table.

We conducted preliminary evaluations on our color based cooperative caching strategy, and we proved its capability of traffic reduction [10]. In this work we extend our idea and we evaluate it under different constraints, such as computational overhead, the number of colors and origin servers, cache capacity, ratio of hybrid caching, as well as under the new proposed routing scheme. Evaluation results show that our colored cooperative caching scheme achieves a performance close to a sub-optimal result calculated by a Genetic Algorithm (GA) while limiting its computational overhead to a few seconds. The proposed routing scheme also reduces more than 30% of traffic compared with the shortest-path routing, and our color distribution scheme can follow changes in access patterns' biases with a few seconds of estimation time.

The rest of the paper is organized as follows. Section 2 describes existing cooperative caching schemes and their routing schemes as well as the video popularity's characteristics. In Sect. 3, we present our color-based caching scheme and a light-weight color management strategy. We also propose a color-based routing algorithm in Sect. 4 utilizing the color information. In Sect. 5, we evaluate the traffic reduction and the computation time overhead while varying the number of colors and origin servers, cache capacity, and the ratio of the hybrid caching using a realistic network topology. Finally, we conclude the paper in Sect. 6.

2. Related Work

2.1 Cooperative Caching Strategies for Reducing Traffic

There are several heuristic approaches for cooperative caching. In an Akamai's CDN [11], they distribute contents within servers in a cluster to increase an effective cache capacity and distribute server load by introducing a hash function. Each server calculates hash values of contents and compares them with server IDs to find appropriate servers to store the contents. Although such strategy could increase effective storage size and distribute server load in a cluster, the effective cache capacity is limited since the authors do not use cooperative servers in different locations. The work [12] adopts a similar approach, but for servers in a ring network in different locations. They distribute contents by calculating hash values with a modulo function, which increases the effective cache capacity. In [13], the authors also propose a hash-based distribution, but for mesh networks. They create several groups of cache servers and distribute contents within a group by associating the hash values and server IDs. Thus they increase effective cache capacity in the network and reduces the traffic to the peering links to CDNs.

Several studies try to find sub-optimal allocations of contents that minimize the traffic size while satisfying several constraints such as throughput, latency, and power consumption [5], [7]. They usually formulate an optimization problem and solve it using an access pattern generated by gathered access logs. However, it is hard to solve such complex problems that include many constraints, since such problems could be transformed to Capacitated Facility Location Problem which is NP-complete [5]. Hence, such optimization problems are usually solved by heuristic approaches such as Genetic Algorithm (GA). Although the produced result could reduce large traffic and distribute server load efficiently, it often takes more than 10 hours of computational overhead even using a PC cluster. Since video accesses continuously change their access patterns in 20–60% every hour [8], such long calculation time often causes mismatches in contents' allocations that generate additional traffic.

2.2 Routing Schemes for Cooperative Caching

The hash-based content distribution approaches [11]–[13] could be used by a simple routing scheme based on hash values. When a server receives a request, it calculates the hash value of the request to find locations of desired contents as well as decide whether to store the content in the server. In fact, such hash-based approaches require a small modification to servers by adding a hash function into their routing scheme. Although they could be used with such a simple routing scheme, they often cause traffic concentrations of popular contents since they do not care about the contents' popularities. Traffic concentration poses many congested links and high-loaded servers that degrade network performance.

When using sub-optimal allocations of contents [5], [7], it is not easy to find cached locations of a desired content without introducing a large size of an additional routing table that associates content IDs and server IDs. Since a sub-optimal contents' allocation changes in time, routing tables must be updated every time when the content allocations change. Moreover, there are a huge amount of contents in the network that makes it difficult to store contents' locations in a small routing table resulting in extra routing overhead.

2.3 Strategies for Following Dynamic Access Patterns

According to the measurement studies of video access patterns, the access patterns continuously change [8] and often make spikes [14] affected by the introduction of new contents, influential news, and viral communications on SNS, such as Twitter and Facebook. A hybrid caching scheme is proposed to follow such access patterns' changes by separating a cache storage into LFU and FIFO caching areas [9]. The LFU area improves the cache hit rate by storing popular contents while the FIFO area follows the changes in access patterns by storing contents recently accessed. However, such hybrid caching schemes do not cooperate with other cache servers that limit the traffic reduction due to the small cache capacity. Prefetching techniques are also proposed [5], [6] that store contents according to future popularities of contents that could be estimated from access logs or empirical rules of VoD service providers. However, the prefetching techniques do not always give a good prediction since video accesses occasionally make a spike due to unpredictable news or viral communications in SNS.

3. Light-Weight Caching Strategy

3.1 Color Tag Based Cooperative Caching

A key factor to reduce network traffic is the use of a combination between content distribution to increase cache capacity, and content duplication to improve cache hit rate. Figure 1 shows the basic concept of our cooperative caching scheme with color tags [10]. Initially, to decide the cache locations in the network, we assign color tags to all cache servers and contents. Each cache server has a tag with a single color, and it stores contents when the server's color matches any of the contents' colors. Since cache servers store different contents based on the colors, effective cache capacity increases, which reduces traffic to the external networks. Servers' color tags are set like the four-color theorem that efficiently distributes contents. The detailed algorithm is explained in [10]. Although it may take a long time to compute an efficient coloration of servers, this overhead can be amortized since the backbone networks do not change their topologies frequently. Contents' color tags are periodically updated to follow changes in the access patterns. Moreover, we apply more than one colors to a tag of popular contents. We aim to distribute traffic load among multiple servers by enhancing hit rates, since popular contents are often the source of traffic concentrations.

The proposed color-based cooperative caching strategy is endorsed with a novel hybrid caching scheme. Figure 2 shows the structure of our hybrid caching scheme with two different caching areas. Each cache server separates its storage area into large colored LFU and small modified LRU areas. The colored LFU area stores color-matching contents to increase effective cache capacity, while the modified LRU area stores recently accessed contents regardless of their color tags to follow rapid changes in access patterns. We adopt a modified LRU algorithm [6] for the modified LRU area. It accepts a jump parameter j, and inserts new contents at j ranks up from the Least Recently Used (LRU) position instead of that of the Most Recently Used (MRU). When cache hits, the modified LRU policy moves the cache-hit content j ranks toward the MRU position. Such behavior achieves higher hit rate than the traditional LRU [6].

The fraction of the colored LFU area and the modified LRU area should be set based on a log analysis or heuristics of VoD service providers. In fact, although a large fraction of the modified LRU area will improve the cache hit rate under a situation with frequent changes in access patterns, such setting degrade the hit rate under static access patterns. Therefore, the fraction of the two cache areas should be carefully set by assuming how frequently the access pattern changes. For example, when a VoD service provider knows that the access pattern changes in 30% every hour based on a preliminary log analysis, they may set the fraction of the hybrid cache capacity to (modified LRU) : (colored LFU) = 10% : 90%, which is known to maintain the hit rate between the periodic updates of color tags based on a preliminary simulation.

3.2 Color Tags Management Algorithm

The contents' tags are periodically updated according to their popularity ranks. The origin server gathers access logs from cache servers and calculates contents' popularities. In particular, server logs contains variations of information that are not necessary to calculate the popularity ranks such as clients' web browsers and protocol versions. Since such in-

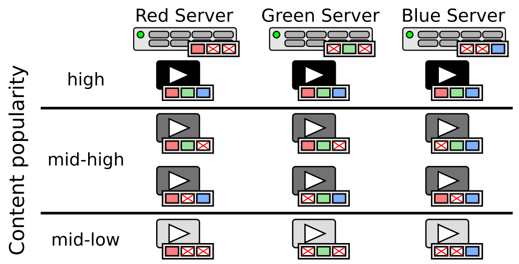

Fig. 1 Example of contents cached in three servers according to their color tags and popularities.

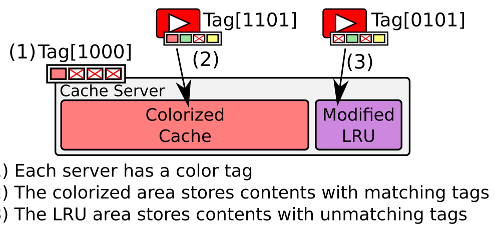
Tag[1101]
Tag[0101]
(1) Tag[1000]
(2)
(3)
Cache Server
Colorized Cache
Modified LRU
(1) Each server has a color tag
(2) The colorized area stores contents with matching tags
(3) The LRU area stores contents with unmatching tags

Fig. 2 Structure of colored hybrid caching strategy and its basic instructions.

Table 1 Popularity classes and corresponding tags in a four-color case.

			Tag's bit patterns			
Popularity class	content popularity ranking	# of colors	R	G	B	Y
high	1–22	4	1	1	1	1
mid-high	23–26	3	1	1	1	0
			1	1	0	1
			1	0	1	1
			0	1	1	1
middle	27-47	2	1	1	0	0
			1	0	1	0
			1	0	0	1
			0	1	1	0
			0	1	0	1
			0	0	1	1
mid-low	48–305	1	1	0	0	0
			0	1	0	0
			0	0	1	0
			0	0	0	1
low	306–	0	0	0	0	0

formation increases an overhead of log gathering process, we assume that each cache server makes a list of tuples <*content_id*, *access_count*> from the access logs, and sends it to the origin server periodically. This list can be generated with O(n) where n is the number of lines of the raw log file. Such logs are sufficiently small compared with the gigabytes of video files since a list of the tuples is only 3.4MB for 100,000 video contents when we assume *content_id* and *access_count* are 32bytes and 4bytes, respectively. After gathering the logs, the contents are grouped into several popularity classes based on their popularity ranks to specify the appropriate number of colors before colorizing them. Table 1 shows an example of popularity classes and the number of colors to be set when we use four colors. The number of contents in each popularity class is decided based on the bias of contents popularity estimated with the access logs. For example, a content with rank 35 will be categorized with a *middle* popularity class and two colors will be assigned to it. In the proposed system, each color tag is described as a bit vector, and the color tags are applied in a cyclic fashion. As a result, the content with popularity rank 35 will have color tag "1001".

We suppose that the users issue their requests with a gamma distriution [15], since it can give us more realistic content accesses than other distribution like Zipf and Weibull distributions. In this assumption, the number of contents in each popularity class depends on the skewness of the curve representing the access probability against the content popularity rank. To decide the number of contents in each popularity class, we use a set of bias parameter k of the gamma distribution and corresponding separator ranks that are the last popularity index in each popularity class. For example, appropriate separator ranks for different k values, when we divide 1000 contents into five popularity classes, are shown in Table 2. The bias parameter k could change in time according to access patterns' changes.

Our goal is to find the optimal separator ranks for a given access pattern, which corresponds to find the adequate number of contents with the adequate popularities that result in a minimum traffic. For example, we suppose to use $N = 4$ colors for 1000 contents, and each cache server allows a maximum storage of $C = 100$ contents to be cached. We initially set the separator ranks equal to [0, 0, 0, 400] which means that first 400 contents belong to the mid-low popularity class, and the rest of the contents are categorized into the low popularity class. Then, we use the iterative calculation shown in Algorithm 1 to find the best set of separator ranks that minimize the traffic. This algorithm searches the best set of separator ranks that minimize traffic by gradually changing each separator rank until no more better separator ranks can be found. The lines 5–16 are corresponding to an iteration to find a better set of separator ranks. The calculate_tail() function calculates a maximum value of $S_{tmp}[N-1]$ that does not overflow the cache capacities. The estimate_traffic() function calculates traffic by calculating

$$T_{est} = \sum_{I}\sum_{J}\sum_{K} e_{ij} p_{ik} y_{ijk} \tag{1}$$

from the cache locations that could be specified by the rank separators, where T_{est} is the total traffic, e_{ij} is a hop count for end-user i to fetch a content from cache server j, p_{ik} is the request probability that end-user i fetches content k, and y_{ijk} is a binary variable indicating whether cache server j storing

Table 2 Separator ranks for each gamma parameter when 1000 contents are classified into five popularity classes.

bias parameter k	separator ranks
0.1	[38, 43, 58, 265]
0.2	[34, 39, 54, 277]
0.3	[30, 34, 55, 285]
0.4	[24, 28, 49, 303]
0.5	[21, 22, 43, 318]
0.6	[15, 15, 42, 332]
0.7	[10, 11, 32, 351]
0.8	[4, 5, 26, 369]
0.9	[0, 0, 10, 392]

Algorithm 1 Iterative calculation for separator ranks

```
 1: Initialization: $S \leftarrow \{0,0,\dots,0\}, S_{prev} \leftarrow \{0,0,\dots,0\}, T_{min} \leftarrow \infty$
 2: $S[N-1] \leftarrow N \times C$
 3: while $S \neq S_{prev}$ do
 4:     $S_{prev} \leftarrow S$
 5:     for $i \leftarrow 0$ to $N-2$ do
 6:         for $v \leftarrow \mathrm{Max}\{0, S[\mathrm{Max}\{1,i\}-1]\}$ to $\mathrm{Min}\{S[i+1], N \times C\}$ do
 7:             $S_{tmp} \leftarrow S$
 8:             $S_{tmp}[i] \leftarrow v$
 9:             $S_{tmp}[N-1] \leftarrow$ calculate_tail($S_{tmp}$)
10:             $T_{est} \leftarrow$ estimate_traffic($S_{tmp}$)
11:             if $T_{est} < T_{min}$ then
12:                 $T_{min} \leftarrow T_{est}$
13:                 $S \leftarrow S_{tmp}$
14:             end if
15:         end for
16:     end for
17: end while
18: return $S$
```

$S, S_{prev}, S_{best}, S_{tmp}$	array of separator ranks, with N elements
T_{min}	minimum traffic in the whole calculation
N	the number of colors
C	the number of contents a cache server can store
T_{est}	traffic size estimated by estimate_traffic()

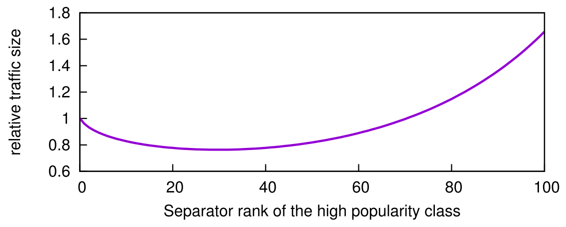

Fig. 3 The first separator rank and traffic size.

content k is the nearest server from end-user i, respectively. This calculation is almost the same with [5], however we do not use bandwidth and latency constraints. The cache locations are uniquely determined from the servers' colors and the contents' colors. The servers' colors are determined by the server colorization algorithm [10], while and the contents' colors are specified according to the color tags management algorithm using separator ranks, which is explained in Sect. 3.2. Figure 3 shows the traffic size variation when we change the first separator rank, or the number of contents categorized in the highest popularity class. This calculation is performed with lines 6–15 in Algorithm 1, and the traffic size in Fig. 3 is corresponding to the T_{est} in the line 9. Figure 3 shows that the optimal rank is about 30. We repeat this procedure to find the separator rank for the other classes. However, once the next separator rank for mid-high popularity class is decided, the separator of the first popularity class may no longer be appropriate. Hence, we repeat the calculation until the separator ranks converge, which corresponds to the while loop starting from the line 3 in Algorithm 1. It is important to mention that the iterative calculation can be performed in advance creating a table like Table 2, since the separator ranks do not change as long as the cache capacity, network topology, and the bias of the content popularity are the same. As a result, the origin server can set appropriate separator ranks with only a small calculation overhead by only estimating the bias parameter k and selecting the corresponding separator ranks from Table 2. We estimate the parameter k that minimizes the Sum of Squared Residuals (SSR) to the measured content popularity from access logs, which requires O(n) where n is the number of contents in the library. Since such relationship between k and SSR forms a convex downward graph, a parameter k that minimizes the SSR can be found using the golden section search [16], which is an algorithm to find an extreme value in unimodal functions with the calculation complexity of O(log(m)) where m is the number of potential k. As a result, we can select a set of gamma parameters from Table 2 with O(n log(m)) calculation.

4. Light-Weight Routing Algorithm Usiung Color Tags

Our color-tag based caching could achieve good traffic reduction even when using the conventional shortest path routing that minimally reaches the origin server. This is due to the fact that a request based on color tags will encounter an appropriate colored cache server on its path. However,

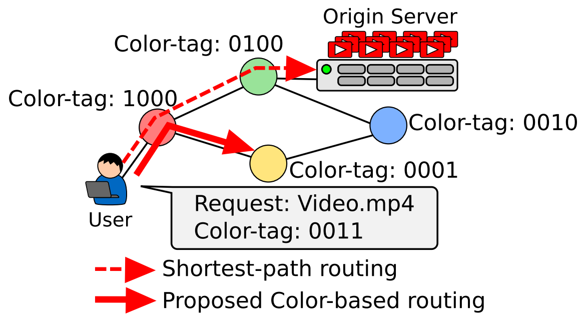

Fig. 4 A proposed color-based routing that finds the nearest cache server with the requesting color-tag.

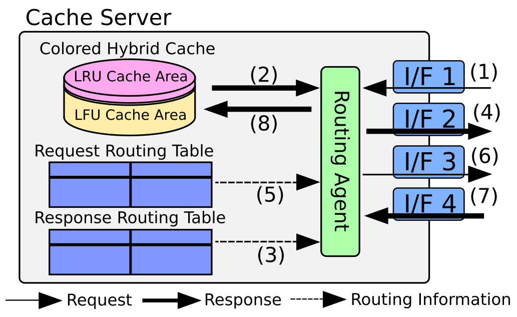

Fig. 5 Structure of the color-tag based cache server.

the nearest server with a matching color may be located in a different direction to the origin server. Hence, we also propose a routing algorithm that utilizes the color information to further reduce traffic by forwarding requests to the nearest servers with matching colors. Figure 4 shows a basic concept of our color-based routing. The user sends a request for a content with an information of content's tag. The tags of contents are preliminarily embedded in the URLs as a request parameter by the origin server, such as http://example.com/video01.mp4?tag=0011. Therefore the users and the cache servers can easily know color tags of the requests. Once a cache server receives the request, it first checks the color information in the request URL, and find the nearest server that matches any of the content's colors. Thus, cache servers do not require a large routing table that associates contents and server IDs.

Figure 5 shows a block diagram of a cache server that enables color based routing based on Algorithm 2. The numbers in the parenthesis in Fig. 5 correspond to those in Algorithm 2. Each cache server has request and response routing tables, LFU/LRU hybrid cache area, routing agent, and network interfaces. (1) When a cache server receives a request, it first checks the cache storage if the requested content is cached locally. When the requested content is cached, (2) the server retrieves the desired content from the cache area by fetchFromCache(). Since the LFU area stores contents that match the server's color, the server searches the LFU area only when the color of the request matches the server's. Otherwise, the server searches the LRU area and responds the cached content if available. After that,

Algorithm 2 Color based routing algorithm

```
 1: if received a $request$ then                                   ▷ (1)
 2:     if desired content is cached then
 3:         $content$ ←fetchFromCache($request$)                   ▷ (2)
 4:         $ipaddr$←sourceAddress($request$)
 5:         $interface$ ← $findResponseInterface$(ipaddr)          ▷ (3)
 6:         sendContent($content, interface$)                      ▷ (4)
 7:     else
 8:         $tag$ ←colorTagOf($request$)
 9:         $interface$ ←findRequestInterface($tag$)               ▷ (5)
10:         sendRequest($request, interface$)                      ▷ (6)
11:     end if
12: end if
13: if received a $response$ then                                  ▷ (7)
14:     $content$ ← $contentOf(response)$
15:     if colorTagOf($response$) & $server\_color$ > 0 then
16:         cacheContent($content$)                                ▷ (8)
17:     end if
18:     $ipaddr$ ← destinationAddress($response$)
19:     $interface$ ← findResponseInterface($ipaddr$)              ▷ (5)
20:     sendContent($content, interface$)                          ▷ (4)
21: end if
```

request	content request message from end-user, containing content ID, color tag, and source IP address
response	response message from the origin or a cache server, containing content data, color tag, and destination IP
interface	network interface to transfer data
server_color	color tag bit of the server like 0010

Table 3 An example of a response routing table.

Destination	Output I/F
10.0.0.0/8	3
20.1.0.0/16	4
20.2.0.0/16	3
30.0.0.0/8	1
default	2

Table 4 An example of a request routing table.

Color mask	Output I/F
0100	3
0001	4
0010	2
1000	1
default	2

(3) the server checks the response routing table to find the appropriate interface to respond the content by findResponseInterface(). Then, (4) the content is sent according to the response routing table by sendContent(). Table 3 shows an example of the response routing table. It is an ordinary routing table that stores network addresses and corresponding output interfaces. If the content is not cached, (5) the server tries to find a network interface to forward by findRequestInterface() and (6) sends the request according to the request routing table by sendRequest(). Table 4 shows an example of the request routing table. It stores information of colors and their corresponding network interface IDs. The color masks are sorted by minimum hops to color-matching servers, which allows the cache server to forward the request to the first color-matching interface. When no color-matching interface is found, it forwards the request to the default interface. (7) When the cache server receives the requested content from the origin or the other cache server, the server checks if the content color matches the server's color and (8) it caches the content and (4) responds to it

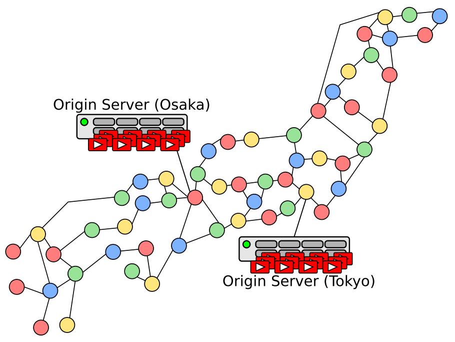

Fig. 6 NTT-like mesh topology [18] and its coloration with four colors.

according to the response routing table. Thus, our routing method utilizes the color tag information in addition to the IP address by the software-based routing algorithm, which enables flexible and efficient routing to forward requests to the nearest color-matching server.

Since our routing algorithm requires only two small color-based routing tables, it does not require large routing overhead. In fact, the number of columns in the request routing table is at most the number of colors + 1. This is considerably small since there are usually millions of contents in the origin's library. Compared to the routing table of Content Centric Networking (CCN) which are based on content categories [17], our routing table is considerably small since only 8 to 16 colors are sufficient to reduce the traffic, as we will see in the evaluation results.

5. Evaluation

5.1 Evaluation Methodology

Evaluations are performed using an NTT-like topology in Japan [18]. Figure 6 shows the adopted topology in our evaluations. The colors of servers are set by the modified version of Welsh-Powell algorithm proposed in [10] in such a way to distribute the server colors in the network. We assume two cases, the first there is a single origin server in Tokyo area only, and a case of two origin servers in Tokyo and Osaka regions, respectively. The two origin servers case is only used for the last experiment, and the rest of experiments are conducted on a single origin server case. We also assume that content access requests are generated from clients interconnected to each node of the topology.

The content requests are generated by the Gamma distribution, which generates more realistic access patterns than other distributions [15] (e.g., Zipf and Weibull distribution). The number of total contents, cache capacity, gamma parameters, and topology information are shown in Table 5. The number of colors applied to each content are decided based on separator ranks that are calculated by Algorithm 1 for all the evaluations. All the calculations are done using a single host with the configuration shown in Table 6. We evaluate normalized traffic size through all the evaluations where a normalized traffic without cache servers is

Table 5 Simulation parameters.

Total content	1000
Cache capacity	100
Popularity distribution	Gamma distribution
Gamma parameter k	0.475
Gamma parameter θ	170.6067
Topology	NTT-like network [18]
Number of servers	55

Table 6 Specification of a computing host.

CPU	Intel Core i7-3930K @3.80GHz
CPU Core	6 physical cores / 12 logical cores
RAM	64GB

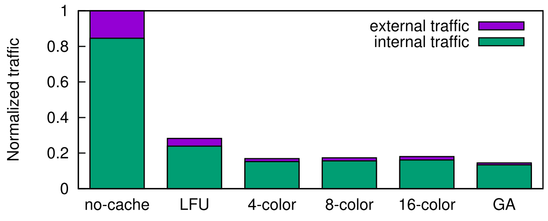

Fig. 7 Normalized traffic on the NTT-like network.

1.0. The cache servers manage their whole storage area with the colored-LFU algorithm (without hybrid caching) in Sects. 5.2, 5.4, 5.6, 5.7, where we evaluate traffic size under static access patterns. The hybrid caching strategy is evaluated in Sect. 5.3, using various storage ratios between 0% and 10% of the modified LRU area.

5.2 Traffic Reduction and Color Management

We evaluate traffic size compared with *no-cache*, LFU, and GA [5] strategies as shown in Fig. 7. Cache servers forward requests and contents by the traditional shortest-path routing [19]. The GA strategy tries to find the best cache locations that minimize traffic, where a set of cache locations forms a gene, and the new cache locations are generated from two genes in the crossover step [5]. The calculation is almost the same with [5] using Eq. (1). Note the external traffic is the traffic between the origin server and a Yellow node in Tokyo area, and the internal traffic is the rest of traffic in the network. *No-cache* is simply a network without any cache servers only for the performance comparison, and all the caching strategies use the conventional shortest path routing. Although LFU is known to be optimal under the static access patterns, it cannot reduce much traffic because of the limited cache capacity due to the absence of cooperative caching. Our color based caching eliminates almost half of the traffic compared to the result of LFU, and when compared to the GA strategy, the difference in hit ratio is lower than 2.3% in a case of 4-color result. The results of 8 and 16 colors increase traffic compared to the 4-color. This is because cache servers cannot forward the request to the nearest color-matching server when we use the shortest-

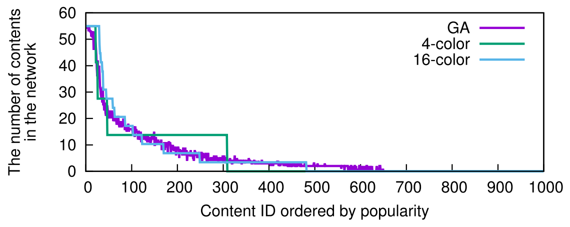

Fig. 8 The number of colorized contents in the network.

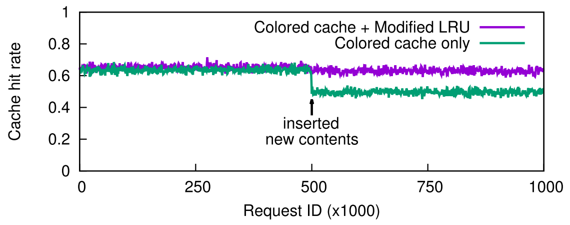

Fig. 9 Transition of cache hit rates when five new popular contents are inserted.

path routing. Cache misses are more likely happened since part of colors may not exist on the shortest-path, resulting to forward the cache miss request to the origin server more when we increase the number of colors. This kind of cache miss can be reduced by a Color-Based Routing which is discussed in Sect. 5.4.

Figure 8 shows the distribution of contents in the network under the different number of colors compared with a sub-optimal result calculated by GA. It is clear that the content distribution with 16-color has a better fit to GA than that of 4-color which should help in the traffic reduction. This result implicates that the color based routing will further reduce traffic utilizing the nearest color-matching servers, and the large number of colors could bring a better balance of content distribution and duplication of popular contents.

5.3 Hybrid Caching Evaluation

We evaluate additional traffic size increased by sudden insertions of popular contents. Our caching scheme follows gradual changes in access patterns by periodic updates of content tags and rapid changes by the hybrid caching scheme. Figure 9 shows the cache hit rates after inserting five popular contents with the lowest popularity (tag=0000), using 10% of modified LRU and 90% of colored LFU areas. While the single use of colored LFU drops its hit rate in 13.9%, our hybrid caching limit the degradation to 2.3%. This is because the no-color LRU area cached newly inserted contents regardless of the color tags.

Figure 10 shows the normalized traffic increase after inserting popular contents against the LRU ratios in a cache area. We can see that a small LRU area effectively reduces the additional traffic. In fact, 5% of LRU area reduced 96.3%, 86.0%, and 71.7% of the additional traffic when inserting of 1, 5, and 10 contents, respectively.

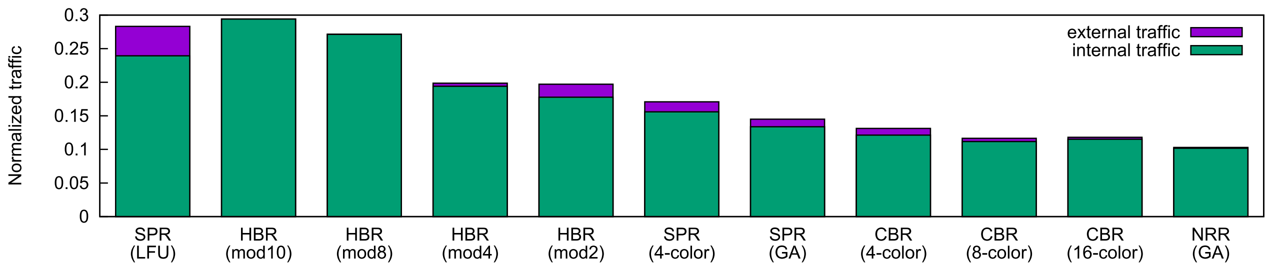

Fig. 11 Traffic reduction under different routing and caching strategies.

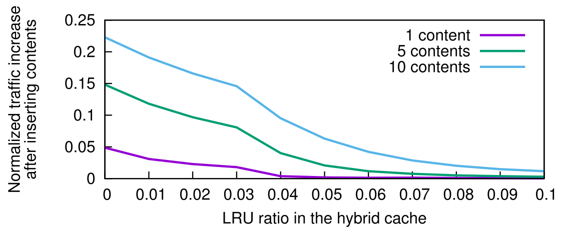

Fig. 10 Traffic increase after inserting popular contents.

Table 7 Computational time in advance for deciding separator ranks.

# of colors	calculation time
4-color	2m2s
8-color	4m1s
16-color	16m2s
GA	763m35s

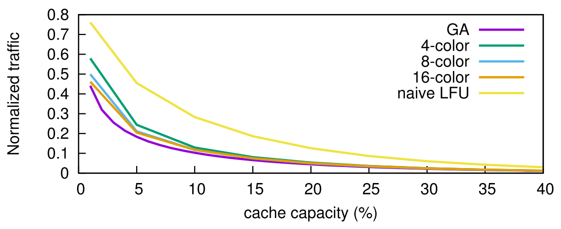

Fig. 12 Traffic size against each server's cache capacity.

5.4 Color Tag Based Routing Scheme

We compare the proposed Color Based Routing (CBR) against the Shortest-Path Routing (SPR) presented in Dijkstra's algorithm [19], the Hash-Based Routing (HBR) proposed in [13], and the Nearest Replica Routing (NRR) which routes requests to the nearest server with the cached content [20]. We numbered cache servers from southwest nodes, and the servers store contents when ($content\_id$ mod N) = ($server\_id$ mod N) in the HBR strategy, where N is the number of colors. Figure 11 shows the normalized traffic under different routing and caching algorithms. The adopted caching strategy is shown in the parenthesis of x-label. CBR (4-color) reduces 23.2% of traffic compared to the SPR (4-color) which proves the benefits that the proposed routing algorithm could bring to the initially proposed caching strategy. CBR (8-color) obtains the best reduction (saves 31.9% of traffic compared to the SPR (4-color)) for aggregated traffic of internal and external links. This is because the large number of colors often slightly increases the number of average hops, since there are a few contents with low popularities in the network. However, CBR (16-color) reduces the most traffic on the external link in our CBR schemes, since the large number of colors could increase the effective cache capacity in the network. The appropriate number of colors could be set by considering several constraints such as traffic cost and power consumption. Moreover, the optimal number of colors may differ according to the target network. For example, a large number of colors could be effective for large networks and small cache capacity. Our CBR schemes also achieve better traffic reduction than the HBR schemes. The HBR (mod10) satisfies all content requests in the internal network, and HBR (mod4) achieves a good traffic reduction among other HBR schemes. However, HBR schemes generate additional internal traffic since users should fetch popular contents from distant cache server resulting to increase the average hops. Consequently, our CBR (8-color) reduces 41.4% of traffic compared to the HBR (mod4).

5.5 Computational Overhead for Color Management

Computational time overhead is considerably reduced for calculating the optimal separator ranks as well as assigning contents' color tags compared with GA. Table 7 shows the computational time for separator ranks for a single gamma parameter. It takes less than 20 minutes while the GA takes more than 750 minutes in our evaluation. The estimation of popularity's bias from access logs takes less than six seconds with a log data with 1,000,000 requests. Moreover, it requires only a few seconds to apply color tags to contents, since our color tags management algorithm only sorts contents by their popularities and applies tags to the contents in a cyclic fashion. These results demonstrate that our color-based caching can follow the changes in access patterns, even with changes in the popularities' skewness.

5.6 Cache Capacity Evaluation

We also evaluate the traffic under several cache capacities. Figure 12 shows the aggregate traffic of the internal and the external links when varying the cache capacity of a single server from 1% to 40% of the whole content library in the origin server. Our colored caching schemes with 4, 8, and

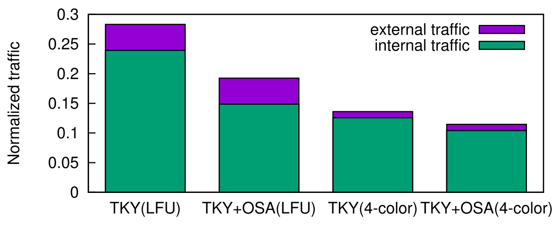

Fig. 13 Origin server locations and normalized traffic size.

16 colors achieve a performance close to the one of GA in all the cache capacity cases. The difference between caching strategies is significant under small cache capacities because of the large penalty of cache misses for popular contents. Moreover, the result of our colored caching scheme gets closer to that of the GA as the cache capacity increases. This is because a large cache capacity stores most of the popular contents and usually diminish the role of caching algorithms.

5.7 Traffic Reduction under Multiple Origin Servers

We also evaluate the traffic size in an environment with multiple origin servers. The origin servers are connected to cache servers in Tokyo (TKY) and Osaka (OSA) as shown in Fig. 6. Figure 13 shows the traffic size of our color-based caching strategy compared with the conventional LFU caching strategy. Firstly, the proposed 4-color based caching reduces internal and external traffic, for both cases of a single and two origin servers, compared to LFU. These are mainly derived because of the cooperating caching effect. Secondly, the internal traffic is reduced for both cases of LFU and 4-color when we increase the number of origin servers. These reductions are caused by reducing the number of average hops to/from nearer origin servers. Finally, the external traffic size is maintained in both LFU and 4-color cases regardless whether the number of origin servers is one or two. This is because the total number of contents stored in cache servers and in-network cache hit ratios are only affected by the underlying cache policies. Hence the proposed 4-color caching is effective not only in the single origin server case but also in the case of two origin servers.

6. Conclusion and Future Work

We have proposed a light-weight cooperative caching scheme introducing simple color tags and their management strategy. The colored caching scheme follows changes in access patterns by a combination use of the colored hybrid caching strategy and the color-based routing algorithm. The evaluation results demonstrate that the proposed caching scheme achieves close to the sub-optimal result calculated by GA, with maintaining high cache hit rates even when access patterns change rapidly. The proposed color management strategy limits the computational overhead to less than six seconds to follow access patterns even with changes in access patterns' biases. The color-based routing further reduces more than 30% of traffic compared to the shortest-path routing by adding two small routing tables to cache servers. Moreover, our evaluation results demonstrated that the proposed caching scheme is effective under various environments. As future work, we plan to extend our coloration scheme to follow access patterns by predicting the future access patterns. It is also interesting to support environments based on a hierarchical topology that consists of access, metro, and longhaul networks, where each video file is separated into a number of chunks and transmitted to the end-user's video player.

Acknowledgements

This work is partly supported by TIS Inc. under a collaborative research project for reducing Internet traffic by efficient utilization of distributed cache servers.

References

[1] "The Zettabyte Era—Trends and Analysis," White Paper, June 2016.
[2] Netflix Inc., "Netflix open connect appliance deployment guide," Feb. 2017. https://openconnect.netflix.com/deploymentguide.pdf, accessed May 1, 2017.
[3] Akamai Technologies, "FAQs: Akamai accelerated network partner (AANP) program," Sept. 2016. https://www.akamai.com/jp/ja/multimedia/documents/akamai/akamai-accelerated-network-partner-aanp-faq.pdf, accessed May 1, 2017.
[4] Google Inc., "Google edge network," https://peering.google.com/#/infrastructure, accessed May 1, 2017.
[5] Z. Li and G. Simon, "In a Telco-CDN, pushing content makes sense," IEEE Trans. Netw. Serv. Manage., vol.10, no.3, pp.300–311, Sept. 2013.
[6] D. De Vleeschauwer and D.C. Robinson, "Optimum caching strategies for a Telco CDN," Bell Labs Technical Journal, vol.16, no.2, pp.115–132, Sept. 2011.
[7] N. Choi, K. Guan, D.C. Kilper, and G. Atkinson, "In-network caching effect on optimal energy consumption in content-centric networking," Proc. 2012 IEEE International Conference on Communications, pp.2889–2894, June 2012.
[8] H. Yu, D. Zheng, B.Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," Proc. 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006, pp.333–344, 2006.
[9] Y. Zhou, L. Chen, C. Yang, and D.M. Chiu, "Video popularity dynamics and its implication for replication," IEEE Trans. Multimed., vol.17, no.8, pp.1273–1285, Aug. 2015.
[10] T. Nakajima, M. Yoshimi, C. Wu, and T. Yoshinaga, "A light-weight content distribution scheme for cooperative caching in Telco-CDNs," Proc. Fourth International Symposium on Computing and Networking (CANDAR '16), pp.126–132, 2016.
[11] B.M. Maggs and R.K. Sitaraman, "Algorithmic nuggets in content delivery," ACM SIGCOMM Computer Communication Review, vol.45, no.3, pp.52–66, July 2015.
[12] Z. Wang, H. Jiang, Y. Sun, J. Li, J. Liu, and E. Dutkiewicz, "A k-coordinated decentralized replica placement algorithm for the ring-based CDN-P2P architecture," Proc. 2010 IEEE Symposium on Computers and Communications, pp.811–816, June 2010.
[13] W. Li, Y. Li, W. Wang, Y. Xin, and Y. Xu, "A collaborative caching scheme with network clustering and hash-routing in CCN," 27th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp.2547–2553, Sept. 2016.
[14] H. Yin, X. Liu, F. Qiu, N. Xia, C. Lin, H. Zhang, V. Sekar, and G. Min, "Inside the bird's nest: Measurements of large-scale live VoD

from the 2008 olympics," Proc. 9th ACM SIGCOMM Conference on Internet Measurement, pp.442–455, 2009.
[15] X. Cheng, J. Liu, and C. Dale, "Understanding the characteristics of Internet short video sharing: A YouTube-based measurement study," IEEE Trans. Multimed., vol.15, no.5, pp.1184–1194, Aug. 2013.
[16] R. Fretcher, Practical Methods of Optimization, John Wiley & Sons, 2000.
[17] S. Hong, B.-J. Lee, C.-M. Yoo, M.-S. Do, and J. Son, "Comparative study of content-centric vs. content delivery networks: Quantitative viewpoints," Proc. 10th International Conference on Future Internet, pp.35–40, 2015.
[18] A. Arteta, B. Barán, and D. Pinto, "Routing and wavelength assignment over WDM optical networks: A comparison between MOACOs and classical approaches," Proc. 4th International IFIP/ACM Latin American Conference on Networking, pp.53–63, 2007.
[19] E.W. Dijkstra, "A note on two problems in connexion with graphs," Numer. Math., vol.1, no.1, pp.269–271, Dec. 1959.
[20] G. Rossini and D. Rossi, "Coupling caching and forwarding: Benefits, analysis, and implementation," Proc. 1st ACM Conference on Information-Centric Networking, pp.127–136, 2014.

Takuma Nakajima received the B.E. degrees from Doshisha University, Japan, in 2013. Since April 2013, he has been studying information network systems at the Graduate School of Information Systems, The University of Electro-Communications, Japan. He received his M.E. degrees in 2014, and is currently working toward Ph.D. degree at the same university.

Masato Yoshimi recieved the B.E., M.E., and Ph.D. degrees from Keio University, Japan, in 2004, 2006, and 2009, respectively. He was Assistant Professor in Doshisha University by 2012. He is currently Assistant Professor in Graduate School of Information Systems, University of Electro-Communications. His research interests include the areas of reconfigurable computing and parallel processing.

Celimuge Wu received his ME degree from the Beijing Institute of Technology, China in 2006, and his Ph.D. degree from The University of Electro-Communications, Japan in 2010. He is currently an associate professor with the Graduate School of Informatics and Engineering, The University of Electro-Communications. His current research interests include computer networks, IoT, and mobile cloud computing. He has been a track co-chair or workshops co-chair for a number of international conferences including IEEE PIMRC 2016, IEEE CCNC 2017, ISNCC 2017 and WICON 2016.

Tsutomu Yoshinaga received his B.E., M.E., and D.E. degrees from Utsunomiya University in 1986, 1988, and 1997, respectively. From 1988 to July 2000, he was a research associate of Faculty of Engineering, Utsunomiya University. Since August 2000, he has been with the Graduate School of Information Systems, UEC, where he is now a professor. His research interests include computer architecture, interconnection networks, and network computing. He is a member of ACM, IEEE, and IPSJ.