IEICE TRANS. INF. & SYST., VOL.E96–D, NO.12 DECEMBER 2013

PAPER Special Section on Parallel and Distributed Computing and Networking

An Approximated Selection Algorithm for Combinations of Content with Virtual Local Server for Traffic Localization in Peer-Assisted Content Delivery Networks

Naoya MAKI[†a)], ***Student Member***, **Ryoichi SHINKUMA**[†b)], ***Senior Member***, ***and*** **Tatsuro TAKAHASHI**[†c)], ***Fellow***

SUMMARY Our prior papers proposed a traffic engineering scheme to further localize traffic in peer-assisted content delivery networks (CDNs). This scheme periodically combines the content files and allows them to obtain the combined content files while keeping the price unchanged from the single-content price in order to induce altruistic clients to download content files that are most likely to contribute to localizing network traffic. However, the selection algorithm in our prior work determined which and when content files should be combined according to the cache states of all clients, which is a kind of unrealistic assumption in terms of computational complexity. This paper proposes a new concept of virtual local server to reduce the computational complexity. We could say that the source server in our mechanism has a virtual caching network inside that reflects the cache states of all clients in the 'actual' caching network and combines content files based on the virtual caching network. In this paper, without determining virtual caching network according to the cache states of all clients, we approximately estimated the virtual caching network from the cache states of the virtual local server of the local domain, which is the aggregated cache state of only altruistic clients in a local domain. Furthermore, we proposed a content selection algorithm based on a virtual caching network. In this paper, we used news life-cycle model as a content model that had the severe changes in cache states, which was a striking instance of dynamic content models. Computer simulations confirmed that our proposed algorithm successfully localized network traffic.

key words: *content delivery network (CDN), peer-assisted network, traffic localization, content combination, content-oriented incentive, news life-cycle*

1. Introduction

Content delivery services such as news, music and software application distribution and video-on-demand have been widely used over the last decade due to the development of communication technology. The delivered volume of content has been increasing since service providers have provided high-definition images and high-fidelity sound. Therefore, the increase in traffic generated by delivering requested content files has become a serious issue and has destabilized the communications of other existing services such as e-mail and Web browsing. Although service providers and network operators have been under increasing pressure to increase bandwidth to solve this problem, it is difficult to deal with traffic that continues to increase without limits. Therefore, it is important to minimize the amount of traffic they generate in their transactions. Peer-assisted content delivery networks (CDNs) have been used as a way of minimizing traffic based on the concept of CDNs [1]–[4]; CDNs distribute storage storing content files replicated from the source server and they direct client requests to the replicas nearest the clients to localize traffic [5]–[7]. Peer-assisted CDNs direct client requests to the nearest replicas as in conventional CDNs, but they can conserve the cost of deploying or borrowing distributed storage since the replicas are stored in the cache of one of millions of clients.

Our prior papers [8]–[10] proposed a traffic engineering scheme for the altruistic clients in peer-assisted CDNs outlined in Fig. 1, in which each altruistic/non-altruistic client independently requests content and replaces its cache using some cache replacement algorithm like first-in/first-out (FIFO). Our scheme used peer-assisted CDNs and periodically combined content files that were most likely to contribute to localizing traffic, while keeping the price equal to that for single content to induce altruistic clients to request them. The main advantage of our approach is that we can expect sustainable contributions from altruistic clients by using sustainable incentives. The source server in our content combination mechanism knows the cache states of all clients in the network, which is a realistic assumption because the service provider should manage every transaction associated with their clients. However, the selection algo-

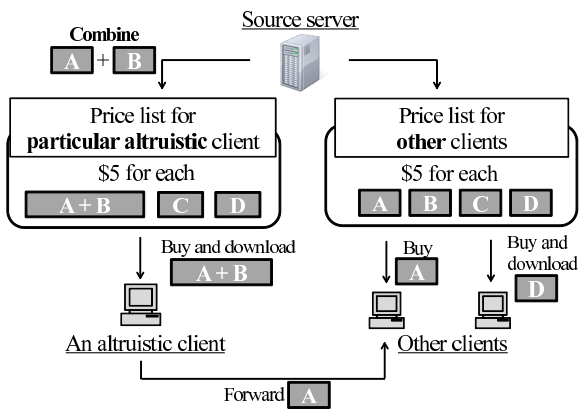


Fig. 1 Example of content combinations. Combination of content A and B is available for altruistic clients for $5 as content. Combination is likely to be requested in local networks.

Manuscript received December 25, 2012.
Manuscript revised May 3, 2013.

[†]The authors are with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
a) E-mail: maki@cube.kuee.kyoto-u.ac.jp
b) E-mail: shinkuma@i.kyoto-u.ac.jp
c) E-mail: ttakahashi@i.kyoto-u.ac.jp

DOI: 10.1587/transinf.E96.D.2684

rithm in our prior work determined which and when content files should be combined according to the cache states of all clients, which is a kind of unrealistic assumption in terms of computational complexity. Therefore, our algorithm had problems with scalability and feasibility.

This paper proposes a new concept of a virtual local server to reduce computational complexity in the selection of combined content. We could say that the source server in our mechanism has a virtual caching network inside that reflects the cache states of all clients in the 'actual' caching network and it combines content files based on the virtual caching network. Without determining virtual caching network according to the cache states of all clients, we approximately estimated the virtual caching network in this research from the cache states of the virtual local server of the local domain, which is the aggregated cache state of only altruistic clients in a local domain. Furthermore, we propose a content selection algorithm based on the virtual caching network. Since the virtual local servers are virtually and distributively deployed, this approach can reduce the amount of computational complexity in two points without incurring any additional infrastructure cost: i) the order of calculations to estimate the virtual caching network, and ii) the frequency of estimates for the virtual caching network.

The four main contributions of this paper are; i) we present the concept of a virtual local server, ii) we propose our content selection algorithm based on an approximated caching network, iii) we present a content model we constructed using news life-cycles, and iv) we reports simulations that confirmed that our approximated algorithm has validity in the news life-cycle model.

The rest of this paper is organized as follows: Sect. 2 describes our assumed service model for the network topology and download of content. Section 3 presents the details on our content combination scheme. Section 4 introduces the concept of a virtual local server and proposes our approximated content selection algorithm. Section 5 presents a content model we constructed by analyzing realistic news sites by assuming request probabilities. Section 6 gives a simulation description and simulation results that validate our proposed algorithm, respectively. Section 7 introduces related work and compares our scheme with this. We conclude the paper in Sect. 8.

2. Our Service Model

There are two main charging structures in general: pay-per-view (PPV) and fixed rate. Clients in PPV pay every time they view content, which does not suit our mechanism because it is not clear how many combined content files there should be so that the revenues of the service provider are not decreased. Clients in our assumed service are charged at a fixed rate and given a fixed number of coupons at each period. They can purchase a content file or a set of combined content files in exchange for a coupon. Even when clients retrieve their purchased content from their own caches, they must use a coupon. A fixed charge is essential in our system so that providing combined content files at a single content price will not reduce the revenues of the service provider. We assumed coupons would be provided frequently enough compared with the average interval between client requests.

Figure 2 outlines the network model we assumed. This is a hierarchical model where the local domains, global domains, and source servers are at the bottom, middle and top layers, respectively. The n_1 and n_2 in Fig. 2 correspond to the number of local domains and the number of clients in each domain.

Figure 3 has the flow for our assumed peer-assisted CDN model. (1) The client first requests content and uses a coupon. (2) Then, the source server makes a transaction on the content charge and redirects the request to a cached location that minimizes traffic; the request is redirected with four priorities: a) the client's own cache, b) the cache at other altruistic clients in the local domain to which the requester belongs, c) the cache at other altruistic clients in the global domain, and d) the source server. (3) When the requester retrieves the required content from the redirected locations of a), b), c), or d), traffic B_0, B_1, B_2, and B_3 are generated ($B_0(=0) < B_1 < B_2 < B_3$). Each client declares that s/he is working as an altruistic or non-altruistic client before joining the service. If the requests are redirected to altruistic clients, they have to forward the requested files.

We have assumed that the system is time-slotted to simplify the discussion in the following section. This supposition is realistic since in the flow of transactions, all client requests are handled by the service provider in a centralized manner in peer-assisted CDNs. Therefore, only a client

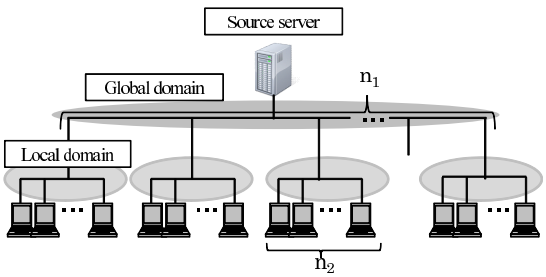


Fig. 2 Network model.

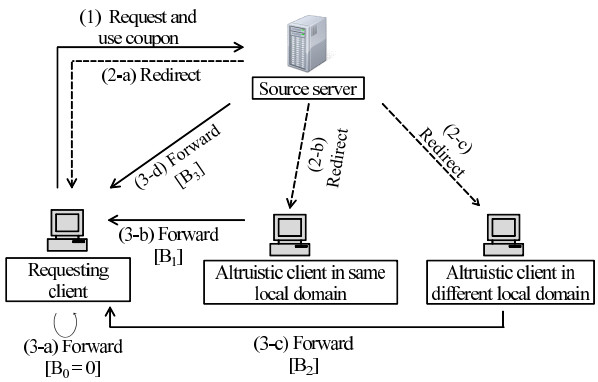


Fig. 3 Peer-assisted CDN model.

is permitted to request and download a content file at each unit-time and the unit-time can be considered to be the average interval between client requests.

3. Our Content Combination Scheme

3.1 Our Incentive Mechanism

As explained in Sect. 1, our basic idea was combined content that would induce altruistic clients to cache content files that were likely to be requested in local networks. In this paper, we have simply considered the request probability of a set of combined content files at time t, $P_{\mathrm{comb}}(\mathrm{t})$, to be:

$$P_{\mathrm{comb}}(t) = \sum_{j \in \mathbb{B}} P_{\mathrm{j}}(t), \tag{1}$$

where $\mathbb{B}$ is a set of the combined content files and $P_{\mathrm{j}}(t)$ is the request probability of content j at time t. Figure 4 has an example of the request probability for combined content files. The request probability of the combination of content B and D equals the sum of the request probabilities of these files. We have left it for future work to find how we will model the request probability of combined content files.

3.2 Problem Formulationx

As we mentioned in Sect. 1, to markedly reduce network traffic, combined content files should be content files that are expected to contribute to localizing traffic generated by transfers. Therefore, we have formulated the optimization of combined content files here.

When the requesting client is an altruistic client, the content file could be a set of combined files. Combined content files under the time-slotted system are not simultaneously provided to multiple altruistic clients. Suppose that a client makes a request in a unit time and altruistic client u is going to request a content file at time t. Altruistic client u obtains a new bundle of combined content files $\mathbb{B}_{(\mathrm{u,t})}$ and removes a set of content files $\mathbb{D}_{(\mathrm{u,t})}$ from its cache $\mathbb{C}_{(\mathrm{u,t})}$ to make space for $\mathbb{B}_{(\mathrm{u,t})}$. To determine the combination of content files, the source server solves the optimization problem written as:

$$\max_{\mathbb{B}_{(\mathrm{u,t})}, \mathbb{D}_{(\mathrm{u,t})}} T\{\zeta_{\mathrm{u}}(S_{\mathrm{t}}) - \zeta_{\mathrm{u}}(S_{\mathrm{t+1}})\} - \eta(\mathbb{B}_{(\mathrm{u,t})}) \tag{2}$$

$$\text{s.t. } S_{\mathrm{t}} \cap S_{\mathrm{t+1}} = (\mathbb{C}_{(1,\mathrm{t})}, \cdots, \mathbb{C}_{(\mathrm{u-1,t})}, \mathbb{C}_{(\mathrm{u+1,t})} \cdots, \mathbb{C}_{(\mathrm{N,t})})$$

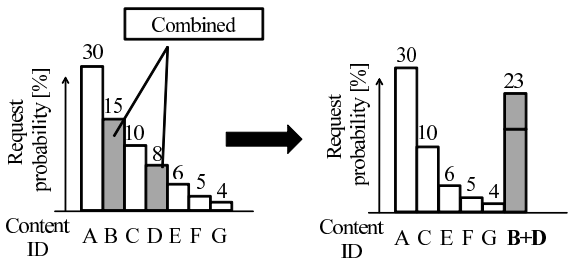


Fig. 4 Example of request probability of combined content.

$$S_{\mathrm{t}} = (\mathbb{C}_{(1,\mathrm{t})} \cdots, \mathbb{C}_{(\mathrm{u,t})}, \cdots, \mathbb{C}_{(\mathrm{N,t})})$$

$$S_{\mathrm{t+1}} = (\mathbb{C}_{(1,\mathrm{t+1})}, \cdots, \mathbb{C}_{(\mathrm{u,t+1})}, \cdots, \mathbb{C}_{(\mathrm{N,t+1})})$$

$$= (\mathbb{C}_{(1,\mathrm{t})}, \cdots, \mathbb{C}_{(\mathrm{u-1,t})}, \mathbb{C}_{(\mathrm{u,t+1})}, \mathbb{C}_{(\mathrm{u+1,t})}, \cdots, \mathbb{C}_{(\mathrm{N,t})})$$

$$\mathbb{C}_{(\mathrm{u,t+1})} = \mathbb{C}_{(\mathrm{u,t})} + \mathbb{B}_{(\mathrm{u,t})} - \mathbb{D}_{(\mathrm{u,t})}$$

$$\zeta_{\mathrm{u}}(S_{\mathrm{t}}) = \sum_{i \in \mathbb{N}} \sum_{j \in \mathbb{Q}} \frac{B_{\mathrm{ij}} P_{\mathrm{j}}(t)}{N}$$

$$\eta(\mathbb{B}_{(\mathrm{u,t})}) = \sum_{j \in \mathbb{B}_{(\mathrm{u,t})}} B_{\mathrm{uj}}(t) - \overline{B_{\mathrm{u}}}(t),$$

where N and $\mathbb{N}$ correspond to the total number of clients and a set of these clients, S_{t} is the state of the cache in the entire network at time t, $\zeta(S_{\mathrm{t}})$ indicates the traffic generated when the cache state is S_{t}, and $\mathbb{Q}$ is the set of all the content files that can be purchased from the service. The $t+1$ means the time just after the cache of client u has been replaced, $\eta(\mathbb{B}_{(\mathrm{u,t})})$ indicates how much traffic is increased by downloading a set of content files $\mathbb{B}_{(\mathrm{u,t})}$ compared with downloading a single file. $B_{\mathrm{ij}}(t)$ represents the traffic generated when client i requests and retrieves content j, and $\overline{B_{\mathrm{u}}}(t)$ indicates the average of traffic generated when client u downloads a single file. As we can see from the definition in Eq. (2), the difference between S_{t} and $S_{\mathrm{t+1}}$ is $\mathbb{B}_{(\mathrm{u,t})} - \mathbb{D}_{(\mathrm{u,t})}$. The $\zeta(S_{\mathrm{t}}) - \zeta(S_{\mathrm{t+1}})$ is how much traffic will be reduced from t to $t+1$ as a result of caching and discarding $\mathbb{B}_{(\mathrm{u,t})}$ and $\mathbb{D}_{(\mathrm{u,t})}$. If the cached content files at clients in the network do not change during period T, $T(\zeta(S_{\mathrm{t}}) - \zeta(S_{\mathrm{t+1}}))$ indicates the amount of reduced traffic during T and T equals T unit times. In fact, cached files in the network change during T. Therefore, we define T as the approximated period where the optimality of the combined and discarded content files is effective. However, downloading $\mathbb{B}_{(\mathrm{u,t})}$ instantaneously generates a large amount of traffic, which is represented as $\eta(\mathbb{B}_{(\mathrm{u,t})})$ in Eq. (2).

3.3 Combined Content Selection

The source server has to know which content files which client has cached to solve Eq. (2). In our prior work, the source server knew the cache states of all clients in the network because the service provider managed every transaction associated with their clients. The selection algorithm for combined content is described in our prior papers [8]–[10].

Next, we will explain our periodic distribution mechanism that uses our content selection algorithm. This mechanism automatically determines T defined in Eq. (2) according to the network cache state and combines content files every period T. Figure 5 is a flowchart of our mechanism, where n and t_{n} correspond to the counters for identifying a retrieved set of combined content files and the time when the set was retrieved. The n is initialized as 0. A client makes a content request at time t. Then, this mechanism determines that the source server should combine content files to localize traffic. Suppose that at time t, the last of the downloads for combined content is that which altruistic client u

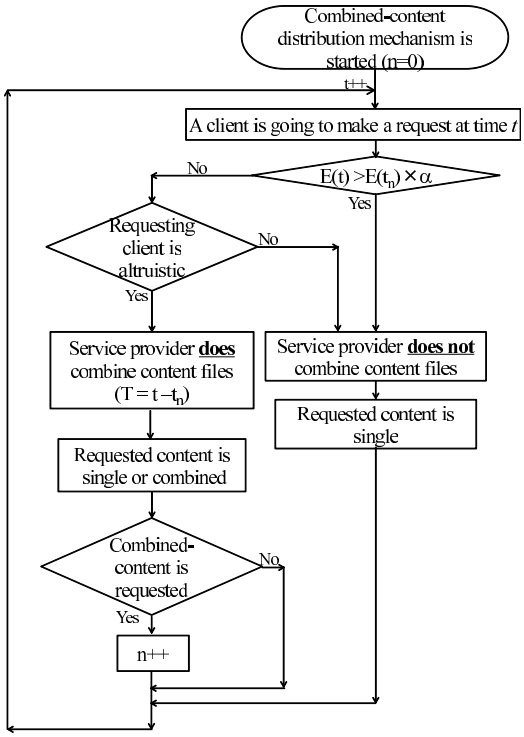


Fig. 5 Flowchart for distribution mechanism.

requested and s/he retrieved combined content at time t_{n}. To determine T for combinations, we introduced the expected traffic reduction of combined content as a metric given by:

$$E(\mathrm{t}) = \sum_{v \in \mathbb{N}} \sum_{k \in \mathbb{B}_{(\mathrm{u,t_n})} \cap \mathbb{C}_{(\mathrm{u,t})}} \frac{P_k(t) \cdot \epsilon^{+}_{\mathrm{vk}}(t)}{N} \tag{3}$$

$$\text{s.t.} \quad \epsilon^{+}_{\mathrm{vk}}(t) = B'_{\mathrm{vk}}(t) - B_{\mathrm{vk}}(t),$$

where $\epsilon^{+}_{\mathrm{vk}}(t)$ and $B'_{\mathrm{vk}}(t)$ mean how much generated traffic is increased and how much traffic is generated when client v requests and obtains content k at time $t+1$ if altruistic client u discards content k at time t, respectively. This mechanism calculates the expected traffic reduction of combined content as a metric to optimize combined content and assesses whether $E(t)/E(t_{\mathrm{n}})$ is still larger than α. If $E(t)/E(t_{\mathrm{n}})$ is larger, the source server does not offer combined content files since we can expect the n-th retrieved combined content to still be working to reduce traffic. If $E(t)/E(t_{\mathrm{n}})$ is smaller, the expected traffic reduction by the n-th downloaded combined content is small because the cache state has been already changed from t_{n}. Therefore, the source server offers the next set of combined content files to further localize traffic by considering the current state of the network cache when a requesting client is altruistic. T is given by:

$$T = t - t_{\mathrm{n}}, \tag{4}$$

where T means the period at which a new set of combined content files is offered. If the offered combined content is requested by the requesting client, n is incremented and t_{n} is updated. If the offered combined content is not requested, T is incremented. The average period where combined content is offered should be obtained from T by using an averaging function. A detailed explanation on the construction of this mechanism is in [10].

3.4 Computational Complexity Problem

As explained in the previous section, to determine $(\mathbb{B}_{(\mathrm{u,t})}, \mathbb{D}_{(\mathrm{u,t})})$ and T: which and when content files should be suitably combined in our mechanism for the caching network, the source server knows the cache states of all clients in the network, which is a realistic assumption because the service provider should manage all transactions associated with their clients. In other words, the source server has a virtual caching network inside as we mentioned in Sect. 1, which reflects the cache states of all clients in the 'actual' caching network and the source server combines content files based on the virtual caching network. Our prior selection algorithm determined a virtual caching network according to the cache states of all clients. However, this method involves high degrees of computational complexity, i.e., i) frequent estimates to determine when content files should be combined, and ii) computational cost $O(N \cdot C)$ in estimating the virtual caching network to determine which and when content files should be combined, where C indicates the extent of cache capacity. Therefore, this approach is kind of unrealistic in terms of computational complexity and becomes a bottleneck in scalability and feasibility.

4. Proposed Method

4.1 Virtual Local Server

This section describes the proposed method and the key concept underlying it is the virtual local server. As we mentioned in Sect. 3.4, the source server has a virtual caching network inside that reflects the cache states of all clients in the 'actual' caching network and the source server combines content files based on the virtual caching network. A virtual local server is introduced into each local domain in the virtual caching network.

Figure 6 is an example of the virtual caching network with virtual local servers. The virtual local server has its own cache state, which reflects the cache states of the clients in its local domain. However, the caching state of the virtual local server is updated according to a simple rule; if either of the altruistic clients in the local domain has a content file in its cache, the virtual local server has the same content in the cache; if no altruistic client in the local domain has a content file in their caches, the virtual local server does not have that content in the cache. In the actual caching network in Fig. 6, there exist only a source server and clients and the source server manages the cache state of the caching network, which is the same assumption as in Figs. 1 to 3. Therefore, actual interactions occur only between the source server and the clients. Virtual local servers are used for the

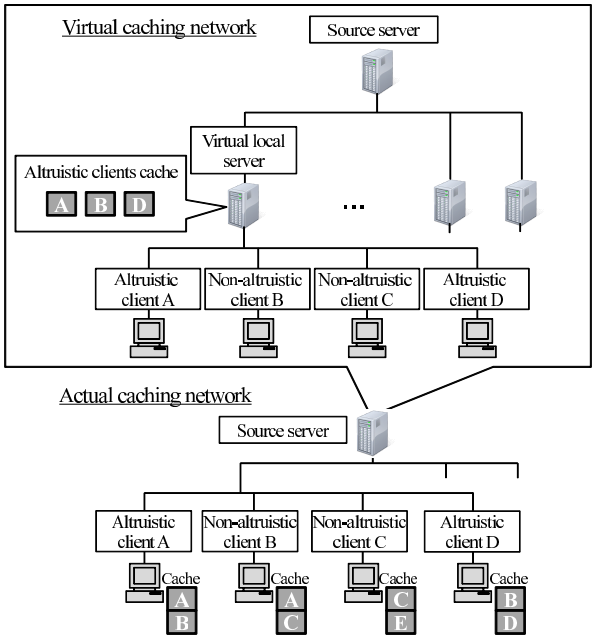


Fig. 6 Example of actual caching network and approximated virtual caching state inside source server. We can assume that all clients cache content A, B, and D because of cache state of virtual local server.

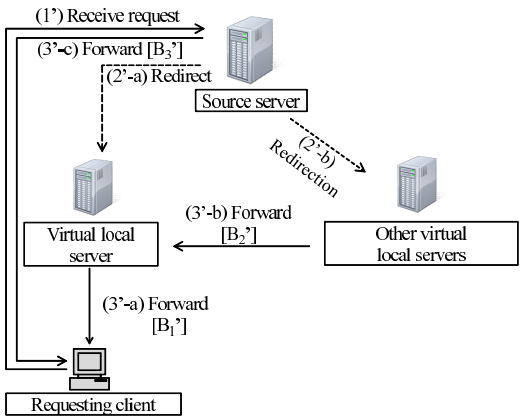


Fig. 7 Virtually approximated service model from source server's standpoint.

source server to manage the network cache state internally. Each altruistic/non-altruistic client requests content and replaces its cache independently using some cache replacement algorithm like FIFO whether virtual local servers are used or not. If the requesting client is altruistic, the cache state of the virtual local server is updated. As presented in the example in Fig. 6, the virtual local server has content A, B, and D in its cache, while it does not have content C and E, which are only cached by non-altruistic clients.

The redirection flow in Fig. 3 has been changed to that in Fig. 7 by introducing virtual local servers as: (1') The source server receives a client's request. (2') Then, this server redirects the request to the source location that can minimize traffic with three priorities: a) the virtual local server in the domain of the requesting client, b) the virtual local server in the other local domains, and c) the source server. (3') When the requester retrieves the required content from the redirected locations of a), b), or c), traffic B'_1, B'_2, and B'_3 are generated ($B'_1 < B'_2 < B'_3$). Note that, since virtual local servers exist only in the virtual caching network, they could not forward and receive actual content files. However, the source server assumes that clients retrieve their requested content files from the virtual local server as described above. Our selection algorithm introduced in Sect. 4.3 was designed based on this assumption.

4.2 Benefits and Drawbacks

As we explained in the previous section, we presented the concept of a virtual local server and a method of approximately creating a virtual caching network with the cache states of virtual local servers. This server can reduce the computational complexity presented in Sect. 3.4; this approach can especially lower the computational cost from $O(N \cdot C)$ to $O(n_1 \cdot C)$. Furthermore, the deployment of virtual local servers does not incur additional infrastructure costs since they are virtually and distributively deployed.

However, the main drawback of the virtual local server is that the aggregated cache states into the cache state of the virtual local server are only altruistic clients', i.e., they are only a part of the clients' cache states. Therefore, we have to approximately estimate the whole virtual caching network from the aggregated cache states of the virtual local servers. Moreover, the expected traffic reduction of combined content is totally overestimated. This is due to the weight of traffic as defined in Sect. 4.1. Overestimating the expected traffic reduction with content that no altruistic client has cached in a local domain is much higher than underestimating it with content that even one altruistic client has cached in a local domain. Therefore, we cannot actually reduce traffic as much as the expected traffic reduction.

4.3 Selection Algorithm

This section briefly describes how we find the sets of combined and discarded content files $\mathbb{B}_{(u,t)}$ and $\mathbb{D}_{(u,t)}$ that satisfy Eq. (2) using three main steps. The generated traffic is calculated from the source server's standpoint and uses B'_1, B'_2, and B'_3 defined in the previous section. Suppose that a client makes a request in unit time and altruistic client u is going to request a content file at time t.

Step 1: Optimization of $\mathbb{B}_{(u,t)}$

Step 1-(a): We calculate the expected traffic reduction during period T with every content j ($j \in \mathbb{Q}$) given by:

$$E^{-}_{l_u j}(t) = \frac{\sum_l \left\{ B^{\overline{C}}_{lj}(t) - B'^{C}_{lj}(t) \right\} n_2 P_j(t)}{N} T - B'_{l_u j}(t), \qquad (5)$$

where $B'^{C}_{lj}(t)$ represents the expected amount of traffic generated when a client joining virtual local server l requests and retrieves content j at time $t+1$ if altruistic client u requests and retrieves content j at time t; $B^{\overline{C}}_{lj}(t)$ represents

if altruistic client u does not request and retrieve content j at time t. Suppose that altruistic client u does not request content j at time t. A client joining virtual local server l requests and retrieves content j at time $t+1$. If the requesting client is delivered from (a) the virtual local server in the domain of the requesting client, $B_{lj}^{'\overline{C}} = B'_1$, (b) the virtual local server in the other local domain, $B_{lj}^{'\overline{C}} = B'_2$, (c) the source server, $B_{lj}^{'\overline{C}} = B'_3$. In the same way, $B_{lj}^{'C}(t)$ can be given when altruistic client u requests content j at time t. Here, $P_{\mathrm{j}}(t)$ is the request probability of content j at time t, and $B'_{\mathrm{l_u j}}(t)$ indicates the traffic generated when client u joining virtual local server l_{u} requests and retrieves content j. $\left\{B_{\mathrm{lj}}^{'\overline{C}}(t) - B_{\mathrm{lj}}^{'C}(t)\right\} n_2$ means the total amount of traffic that is expected to be reduced by every client's request in virtual local server l at time $t+1$ if altruistic client u requests and retrieves content j at time t. Because the cache state is identical for clients in the local domain, we multiply n_2 as approximated in Sect. 4.1. The first term is used to calculate the amount of traffic that is expected to be reduced during period T if the amount of traffic that is expected to be reduced at time $t+1$ remains unchanged during T. Subtracting $B'_{\mathrm{l_u j}}(t)$ in Eq. (5) takes into consideration the fact that, as we increase the number of combined content files, more instantaneous traffic is generated.

Step 1-(b): We score every content $P_{\mathrm{j}}(t)E^{-}_{\mathrm{l_u j}}(t)$ and sort them in descending order. This is because $P_{\mathrm{comb}}(t)$ defined in Eq. (1) should also be considered because content j will not be effective if it is not actually requested and cached by altruistic client u. This step works to increase the request probability of combined content files that will reduce a large amount of traffic at time $t+1$.

As explained in Sect. 4.2, as the request probability from Eq. (5) is high, the expected traffic reduction is greatly overestimated. Therefore, our proposed algorithm is likely to make virtual local servers obtain or retain content files that are very popular.

Step 2: Optimization of $\mathbb{D}_{(\mathrm{u,t})}$

Step 2-(a): Next, we calculate the expected traffic increase during period T when a virtual local server discards content k given by:

$$E^{+}_{\mathrm{l_u k}}(t) = \frac{\sum_l \left\{B^{'+}_{\mathrm{lk}}(t) - B^{'-}_{\mathrm{lk}}(t)\right\} n_2 P_{\mathrm{k}}(t)}{N} T, \tag{6}$$

where $B^{'+}_{\mathrm{lk}}(t)$ and $B^{'-}_{\mathrm{lk}}(t)$ represent the expected amount of traffic generated when a client joining virtual local server l requests and retrieves content k at time $t+1$ if altruistic client u does or does not discard content k at time t, respectively. Similarly to $B_{lj}^{'\overline{C}}$, $B^{'+}_{lk}(t)$ can be given when altruistic client u discards content j at time t. $B^{'-}_{lk}(t)$ can be given when altruistic client u does not discard content j at time t. However, this only happens when no altruistic clients except requesting client u cache content k and client u is going to discard content k ($k \in \mathbb{C}_{(\mathrm{u,t})}$) at time t. Similarly to that in Eq. (5), Eq. (6) is used to calculate the amount of virtual traffic that is expected to be increased during period T when altruistic client u discards content k at time t.

Step 2-(b): We score the cached content of altruistic client u $P_{\mathrm{k}}(t)E^{+}_{\mathrm{l_u k}}(t)$ and sort it in ascending order. $P_{\mathrm{k}}(t)$ needs to be considered because, in our model described in Sect. 2, altruistic client u can request the content cached in his or her cache space; we can increase $P_{\mathrm{comb}}(t)$ by attaching content already cached at client u with larger $P_{\mathrm{k}}(t)$ to the combined files while discarding content with smaller $P_{\mathrm{k}}(t)$.

Step 3: Optimization of Eq. (2)

We can simplify the optimization of Eq. (2) to the following discrete optimization problem as a function of x, which represents the number of content files included in $\mathbb{B}_{(\mathrm{u,t})}$:

$$\max_x G_{\mathrm{comb}}(t) P_{\mathrm{comb}}(t) \tag{7}$$

$$\text{s.t.} \quad G_{\mathrm{comb}}(t) = \sum_{g=1}^{x} \left(E^{-}_{\mathrm{l_u b_g}}(t) - E^{+}_{\mathrm{l_u d_g}}(t)\right) + B'_{\mathrm{l_u b_1}}(t)$$

$$P_{\mathrm{comb}}(t) = \sum_{g=1}^{x} P_{\mathrm{b_g}}(t) + \sum_{h=C-x}^{C} P_{\mathrm{d_h}}(t),$$

where $G_{\mathrm{comb}}(t)$ represents the amount of traffic reduced by combined content files, b_{g} is the identification number of the content with the g-th largest $P_{\mathrm{j}}(t)E^{-}_{\mathrm{l_u j}}(t)$, d_{h} is the identification number of content with the h-th smallest $P_{\mathrm{k}}(t)E^{+}_{\mathrm{l_u k}}(t)$. Here, C is the cache capacity of altruistic client u. We determine the combination of $\mathbb{B}_{(\mathrm{u,t})}$ and $\mathbb{D}_{(\mathrm{u,t})}$ on the basis of Eq. (7) as:

$$\mathbb{B}_{(\mathrm{u,t})} = \mathbb{D}_{(\mathrm{u,t})} = \phi \quad (if \;\; G_{\mathrm{comb}} P_{\mathrm{comb}} < 0)$$
$$\mathbb{B}_{(\mathrm{u,t})} = (b_1, b_2, \cdots b_{\mathrm{x}}), \mathbb{D}_{(\mathrm{u,t})} = (d_1, d_2, \cdots d_{\mathrm{x}}) \quad (else).$$

Since b_{g} and d_{h} are sorted, $G_{\mathrm{comb}}(t)P_{\mathrm{comb}}(t)$ becomes a convex function. Therefore, we can easily solve the discrete optimization problem and obtain the optimal number of content files to be combined.

When we use a virtual local server in our periodic distribution mechanism, the expected traffic reduction of combined content, as we defined in Eq. (3), can be approximated as.

$$E(\mathrm{t}) = \sum_{k \in \mathbb{B}_{(\mathrm{u,t_n})} \cap \mathbb{C}_{(\mathrm{u,t})}} P_k(t) E^{+}_{\mathrm{l_u k}}(t). \tag{8}$$

5. News Content Model

We evaluated the performance of our scheme in our prior work [8]–[10] with only a static content model, which did not change the request probability or generate new content files. In reality, the content model should be dynamic. Here, we used a news life-cycle as the content model as it is a striking instance of dynamic content models. Since many newsworthy events are generated in a day [11]–[13] and instantaneous reports are important for news articles [14], [15], the frequency of generation of news content is higher and the change in the popularity of content is much more frequent those for entertainment.

5.1 Number of Generated News Articles

We analyzed news articles treated as headlines on the top page of "Yahoo! Japan News" from July 2004 to July 2012 [11] to study the number of news articles generated in a day. Yahoo! Japan is one of the largest portal sites in Japan and Yahoo! Japan News is a headline service that distributes topical news that newspaper publishing companies provide. News content distributed as headlines are selected by how many users are interested in them. Therefore, we can expect all articles listed on the top page as headlines to have been accessed more than a certain number of times.

Figure 8 plots the total number of days as a function of the number of news articles at the top page of Yahoo! Japan News generated in a day. The minimum and maximum number of news articles generated in a day in Fig. 8 have been 20 for the former and 82 for the latter in the last eight years. Furthermore, the range from 50 to 64 includes the top three ranges of the number of news articles generated in a day. We found that the average number of news articles generated in a day was 51.2 and their dispersion was 11.1; if news articles are generated at equal intervals, they are generated every 28.1 minutes.

5.2 Request Probability

It is generally well-known that the request probability of entertainment content such as music and movies follows Zipf's law [16]. The request probability of content that has the i-th highest popularity is:

$$P_{\mathrm{i}} = \frac{\frac{1}{i}}{\sum_{j\in\mathbb{Q}} \frac{1}{j}}. \tag{9}$$

However, unlike entertainment content, clients prefer real-time news articles to old ones since immediacy is important for news articles. Therefore, the rate of decline in the request probability of news content is sharper than that in entertainment content and does not follow Zipf's law. Some conventional research has analyzed and reported the life cycle of news articles [14]–[18]. Most of them can be approximated by monotonic decrease models. In this paper, we can approximate the request probability of news contents by a power law:

$$P_{\mathrm{i}} = \frac{a^{bi}}{\sum_{j\in\mathbb{Q}} a^{bj}} \quad (0 < a < 1,\ \ 0 < b < 1). \tag{10}$$

Figure 9 compares request probabilities following Zipf's law and a power law. As we can see from Fig. 9, the request probabilities following the power law have more rapidly declined than Zipf's law. Furthermore, the power law has a feature that the rate of decline of request probability decreased as parameter a decreased or parameter b increased as seen in Eq. (10).

As we mentioned in Sect. 5.1, headlines, which are likely to attract more than a certain number of accesses, have comparatively higher initial request probabilities than other news articles. Therefore, every news content file on headlines can be assumed to similarly trace the transition of request probability and can be estimated in advance. In this paper, the popularity of all news content files are lowered one rank in Eq. (10) when a new content file is generated while the popularity of newly generated content is set at the highest rank. This is because recent news content that has real time information is mostly accessed. Figure 10 has an example of the change in request probabilities when a news content file is newly generated. When content H is generated at time $t+1$, all content except for content H is lowered

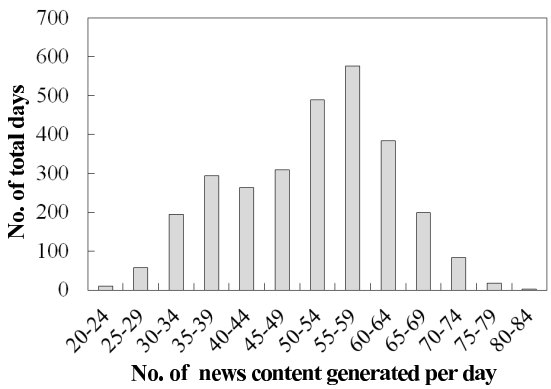


Fig. 8 Total number of days as function of number of news articles generated on the top page of Yahoo! Japan News per day.

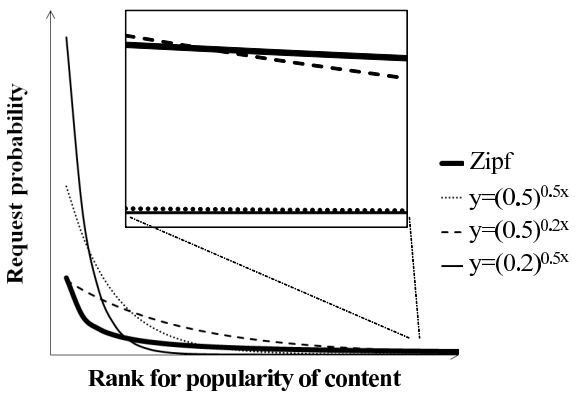


Fig. 9 Comparison of request probabilities following Zipf's law and power law.

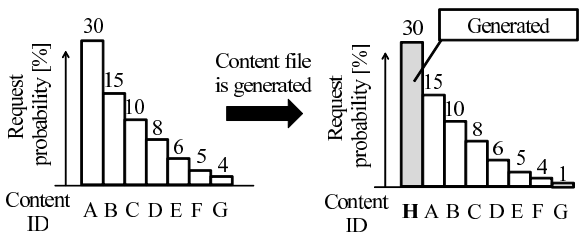


Fig. 10 Example of change in request probabilities when news article is newly generated.

one rank while content H reaches the highest rank. Then, the change in the request probability of content is:

$$P_{\mathrm{A}}(t) = P_1, P_{\mathrm{B}}(t) = P_2, P_{\mathrm{C}}(t) = P_3, \cdots$$
$$\rightarrow P_{\mathrm{H}}(t+1) = P_1, P_{\mathrm{A}}(t+1) = P_2, P_{\mathrm{B}}(t+1) = P_3 \cdots.$$

6. Simulation Descriptions

We used simulations to verify that our scheme, which used the approximated selection algorithm proposed in Sect. 4 could effectively reduce the amount of network traffic. The parameters we used are listed in Table 1. We used the scale of distribution in our prior work [8]–[10]. The generation interval of news content was found from the results of analysis from Yahoo! Japan News in Sect. 5.1. Furthermore, we used the number of news articles placed on the top page of Yahoo! Japan as the number of news articles a client views per day. In this simulations, we set B'_1 equal to B_0. It depends on the number of altruistic clients in the domain and the number of cached files of the client how likely a client retrieves the requested content from other altruistic client in the same local domain or from its own cache. However, the traffic weights B_3 and B_2 are much larger than B_1 and B_0, i.e., $B_3 > B_2 \gg B_1 \sim B_0$. Therefore, it would not change any results essentially if we use B_1 for B'_1 instead of B_0.

We observed how much more traffic was reduced with peer-assisted CDNs by using our scheme than when it was not used. The only difference between these cases is that our scheme periodically provided combined content files to control the request probabilities of altruistic clients and a set of combined content files could be made available by only using a coupon. We used the evaluation metric defined as:

$$\Psi = \frac{\tau^{\overline{C}} - \tau^{C}}{\tau^{\overline{C}}}, \tag{11}$$

where τ^{C} and $\tau^{\overline{C}}$ correspond to how much traffic is generated in the entire network when we use or do not use our scheme.

We also observed how many content files were combined and from which cached location they were retrieved from by an altruistic client, i.e., the cache of the altruistic client, the local domain, the global domain, or the source server.

Table 1 Our parameters used in simulation.

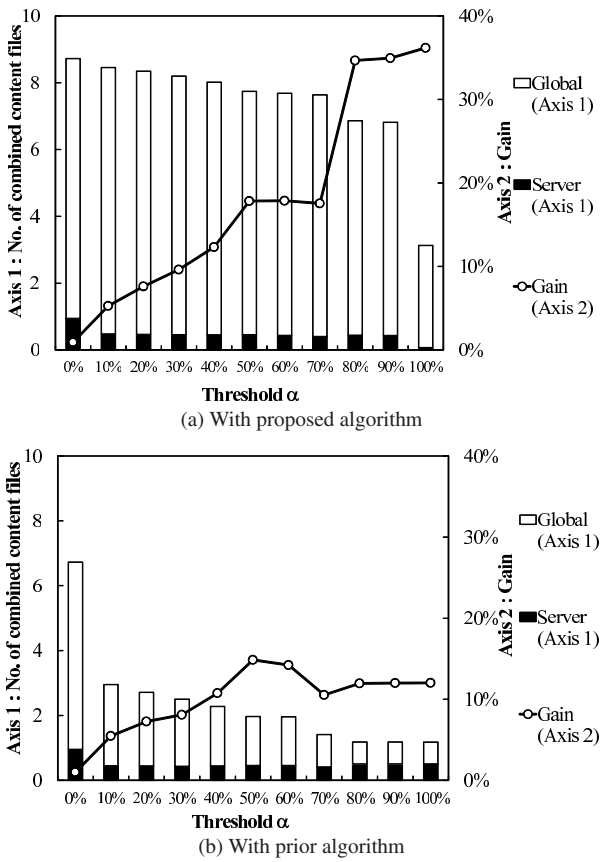
No. of purchasable content files	1000
No. of local domains (n_1)	50
No. of clients in each local domain (n_2)	40
Total no. of clients (N)	2000
Ratio of altruistic clients (r)	10%
Cache capacity at each client (C)	10
Cache replacement algorithm	first-in/first-out (FIFO)
Traffic weight (B_3,B_2,B_1,B_0)	(2000,50,1,0)
Virtual traffic weight (B'_3,B'_2,B'_1)	(B_3,B_2,B_0)
Interval for content generation (I)	30min
Parameters a and b in Eq. (10)	0.5
Threshold (α)	90%
No. of news articles every client views per day	8
Observation period	25 days

6.1 Dependence on Cache Characteristics

6.1.1 vs. Threshold α

Let us first analyze the performance of our scheme for different thresholds α defined in Sect. 3.3. Figure 11 (a) plots the gain defined in Eq. (11) and the number of combined content files as a function of threshold α with our proposed algorithm. "Global" and "server" mean the number of combined content files from the global domain and the source server in Fig. 2. We can see that our proposed algorithm can effectively reduce network traffic. Here, let us compare how much traffic our approximated selection algorithm proposed in Sect. 4 generated with the our prior selection algorithm presented in [8]–[10].

Figure 11 (b) plots the gain and the total number of combined content files with our prior algorithm. As we can see from Figs. 11 (a) and 11 (b), the gain and the total number of combined content files with the proposed algorithm basically became larger than those with the prior algorithm.



(a) With proposed algorithm

(b) With prior algorithm

Fig. 11 Number of combined content files and gain as function of threshold α. These graphs compare evaluation metrics between (a) our proposed and (b) prior selection algorithm. Bar graphs indicate number of combined content files, while line graphs indicate gain.

Our prior algorithm determines which content files should be combined by the network cache state at time t. Therefore, the optimality of combined content files at time t is decreased as the time passes. If the combined content files contain popular content files, the optimality can be sustained because popular content files are not likely to disappear from the network.

Our proposed algorithm also uses the network cache state at time t to choose combined content files. However, unlike our prior algorithm, it considers only the cache state of altruistic clients. Suppose that all non-altruistic clients in a local domain have a specific popular content file in their caches. Our prior algorithm tends not to put the specific content in the combined content because we could not expect a large amount of traffic reduction in the local domain. However, our proposed algorithm does that because it does not consider the cache state of non-altruistic clients and still expect the popular content would reduce a large amount of traffic. Thus, our proposed algorithm caches popular content files more likely than our prior algorithm. If we discuss only the optimality at time t, our prior algorithm is better. However, since, as we explained above, cached popular content files results in the sustained optimality after time t, our proposed algorithm works better.

We have plotted the theoretical request probability of combined content defined in Eq. (1) as a function of threshold α in Fig. 12 where we can see that the theoretical request probability of combined content with our proposed algorithm was actually higher than that with our prior algorithm. This is also why our proposed algorithm further reduced traffic in Fig. 11 (a).

Next, we will discuss what we obtained by varying threshold α. We can see that the gain increased as α increased. This is because of the period where combined content was distributed. In reality, Fig. 12 plots the value of T defined in Eq. (2) as a function of threshold α, and we can see that T increased as α decreased. Therefore, although more content files were combined as α decreased, our scheme was less likely to combine content files while the expected traffic reduction of previously downloaded combined content largely decreased. Setting a large α means the optimality of combined content largely decreases even if cached files are just partly replaced.

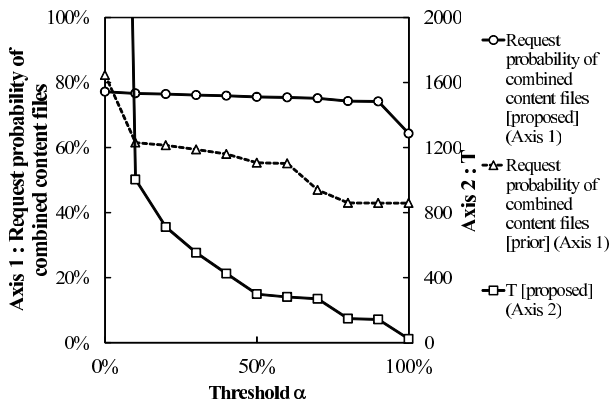


Fig. 12 Theoretical request probability of combined content and T as function of threshold α. On Axis 1, solid and dashed lines correspond to theoretical request probability of combined content files with our proposed and prior selection algorithm. On Axis 2, line indicates T with our proposed algorithm.

6.1.2 vs. Cache Replacement Algorithm

Figures 11 (a) and 13 compare the gains and the number of combined content files between FIFO and the least-recently used (LRU) algorithm as a function of threshold α. The line of gain in Fig. 13 and the bar graph for the number of combined content files in LRU coincide with those in FIFO. This is because although LRU operates so that popular content files are more likely cached, LRU behaves similarly to FIFO since new and old content files are much more likely to be requested and discarded in the news life-cycle model.

6.1.3 vs. Ratio of Altruistic Clients r

Figure 14 plots the gain and the number of combined content files as a function of the ratio of altruistic clients r. We can see here that our proposed algorithm reduces traf-

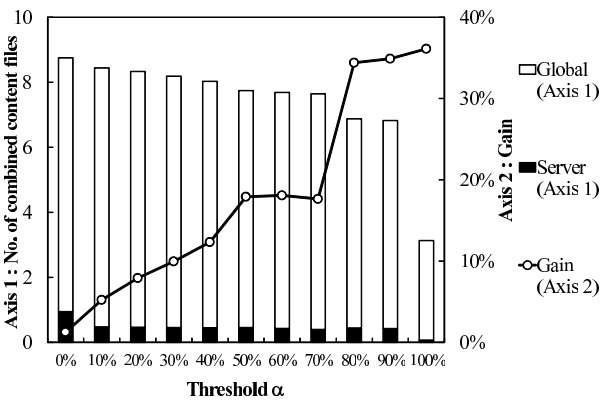


Fig. 13 Number of combined content files and gain as function of threshold α when we used LRU algorithm. Bar graphs indicate number of combined content files, while line graphs indicate gain.

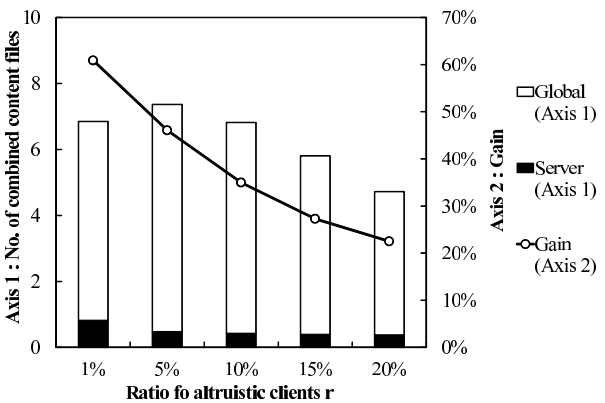


Fig. 14 Number of combined content files and gain as function of ratio of altruistic clients r. Bar graph indicates number of combined content files, while line graph indicates gain.

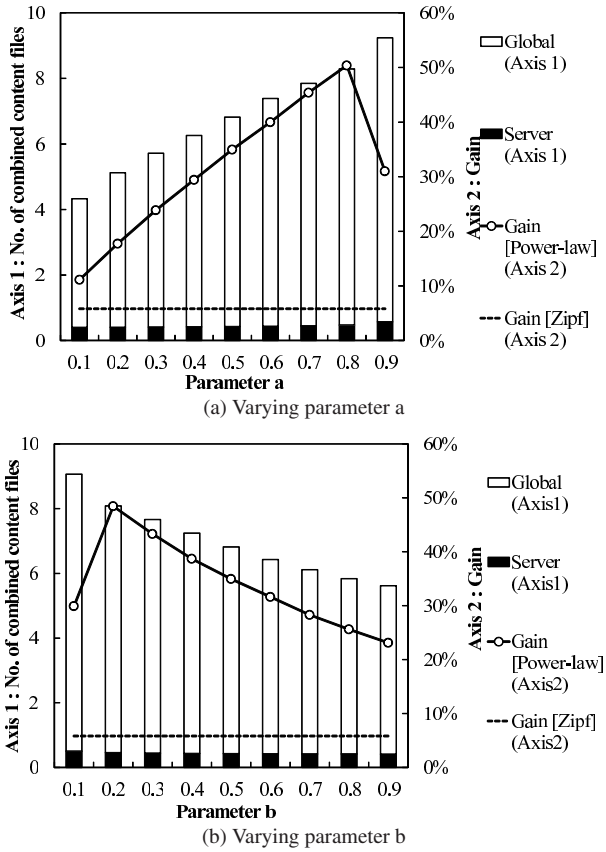


Fig. 15 Number of combined content files and gain as function of parameter of request probability (a) a and (b) b. Bar graphs indicate number of combined content files, while line graphs indicate gain. Solid and dashed lines correspond to gains when request probability follows power law and Zipf's law.

fic by 60.9% when $r = 1\%$. As the ratio of altruistic clients increases, traffic has already been more localized without our scheme since many content files requested by clients are found in the caches of altruistic clients. Therefore, the gain and the number of combined content files decrease.

6.2 Dependence on Popularity Model

6.2.1 vs. Parameters of Request Probability a and b

Figures 15 (a) and 15 (b) plot the gains and the number of combined content files as a function of the parameters of request probability (a) a and (b) b defined in Eq. (10), respectively.

Let us first discuss what we obtained by varying parameter a from Fig. 15 (a). We can see that the gain peaks at $a = 0.8$. When a is less than 0.8, the gain simply decreases. This is because, as mentioned in Sect. 5.2, since the distribution of request probability becomes more steeply inclined as a decreases, only a smaller fraction of a set of purchasable content files is requested by clients. Therefore, the number of combined content files also decreases. However, when a exceeds 0.8, the gain decreases. When a increases, clients begin to request various content files and the number of

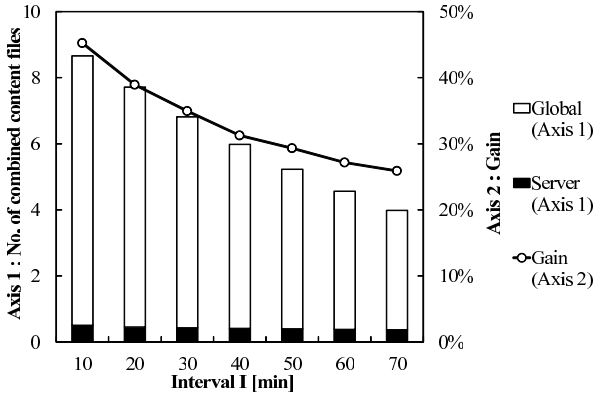


Fig. 16 Number of combined content files and gain as function of interval of content generation I. Bar graph indicates number of combined content files, while line graph indicates gain.

combined content files increases. However, clients easily request content files not included in the combined content files selected by our scheme because the cache capacity limits the number of combined content files. As a target for comparison, we have also plotted the gain in Fig. 15 (a) when the request probability follows Zipf's law defined in Eq. (9); the distribution of Zipf's law is more gently inclined than that of the power law described in Fig. 9. The gain when the request probability follows Zipf's law is much smaller than that when the request probability follows the power law.

We can see a similar trend in Figs. 15 (a) and 15 (b), and Fig. 15 (b) peaks at $b = 0.2$. The distribution of request probability becomes more steeply inclined as b increases. Therefore, as mentioned in the explanation of Fig. 15 (a), the number of combined content files decreases as parameter b is increased, while clients more easily request content files not included in the combined content files selected by our scheme as parameter b is decreased. This is why the gain has a peak.

6.2.2 vs. Intervals for Content Generation I

Figure 16 plots the gain and the number of combined content files as a function of the intervals of content generation I. We can see here that our proposed algorithm successfully reduces large amounts of traffic when the intervals for content generation are short; it can reduce traffic by 45.2% when new content files are generated every 10 min. Generating new content files during long intervals means that the rate of change in request probability is low. Therefore, popular content files have already been cached locally. This is why gain and the number of combined content files decreased when I became long.

7. Related Work

Content placement in peer-to-peer (P2P) networks and CDNs are long-standing and well-studied problems; besides the papers we cited in our prior papers, [19]–[22] treated these problems. In peer-assisted CDNs, unlike P2P net-

works and CDNs, a central entity can attempt to control cache placement [23], but clients may refuse to cache directed content files because peer-assisted CDNs owe clients storage resources. As we explained in Sect. 3.1, our approach does not directly control caches at clients but only induces altruistic clients to cache required content to localize traffic.

Since we assumed a paid service in this paper, all clients have good reasons not to contribute to the service as content servers. Therefore, we should expect a limited number of altruistic clients to contribute to the service [24]. Our purposed scheme was simply to induce altruistic clients to request specific content that would likely be requested on local networks to reduce traffic. The form of the incentives was another factor in which we were interested. It is mathematically unclear how much a certain level of improved service quality or monetary gain would increase the probability that free riders would contribute to networks [25], [26]. Our incentive by combining content, which was explained in Sect. 3.1, is straightforward. As we explained in Sect. 2, since clients are charged at a fixed rate to obtain coupons for every period and the combined content is just an electronic copy of the original file, we can ignore the source of the incentive or reward once the service providers obtain Internet-delivery rights. Distribution with packed content files has been used as a business model to motivate consumers to make purchases in pay-per-view services [27], [28].

8. Conclusion

This paper proposed a new concept of a virtual local server to reduce computational complexity in the selection of combined content. In our content combination mechanism, we could say that the source server has a virtual caching network inside that reflects the cache states of all clients in the 'actual' caching network and combines content files based on the virtual caching network. Without determining virtual caching network according to the cache states of all clients, we approximately estimated the virtual caching network in this research from the cache states of the virtual local server of the local domain, which is the aggregated cache state of only altruistic clients in a local domain. Furthermore, we proposed a content selection algorithm that determined which and when content files should be combined according to the virtual caching network. Our proposed algorithm allowed us to decrease the computational complexity; i) the frequency of estimates of the virtual caching network and ii) the computational cost in estimating the virtual caching network. We constructed a news life-cycle model in this research as a content model that had major changes to cache states, which was a striking instance of dynamic content models. Our computer simulations confirmed that our proposed algorithm increased the request probability of combined content by overestimating the expected traffic reduction of its content files which were required for the localization. Furthermore, our proposed algorithm could reduce traffic by about 40% even if the service environment changed.

References

[1] T. Mori, N. Kamiyama, S. Harada, H. Hasegawa, and R. Kawahara, "Improving deployability of peer-assisted CDN platform with incentive," Proc. IEEE GLOBECOM'09, pp.1–7, Honolulu, Nov. 2009.
[2] D. Xu, S. Kulkarni, C. Rosenberg, and H. Chai, "Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution," Mutimedia Systems, vol.11, no.4, pp.383–399, March 2006.
[3] C. Huang, A. Wang, J. Li, and K.W. Ross, "Understanding hybrid CDN-P2P: why Limelight needs its own red swoosh," Proc. 18th NOSSDAV'08, pp.75–80, Braunschweig, Germany, May 2008.
[4] "BitTorrent DNA," http://www.bittorrent.com/dna
[5] G. Peng, "CDN: Content distribution network," Dept. Computer Science, State Univ. of New York, New York, Tech. Rep. TR-125, 2003.
[6] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," Commum. ACM, vol.49, no.1, pp.101–106, 2006.
[7] "Akamai," http://www.akamai.com/index.html?intl=1
[8] N. Maki, T. Nishio, R. Shinkuma, T. Mori, N. Kamiyama, R. Kawahara, and T. Takahashi, "Traffic engineering of peer-assisted content delivery network with content-oriented incentive mechanism," IEICE Trans. Inf. & Syst., vol.E95-D, no.12, pp.2860–2869, Dec. 2012.
[9] N. Maki, T. Nishio, R. Shinkuma, T. Mori, N. Kamiyama, R. Kawahara, and T. Takahashi, "Expected traffic reduction by content-oriented incentive in peer-assisted content delivery networks," Proc. International Conference on Information Networking (ICOIN 2013), Bangkok, Thai, pp.450–455, Jan. 2013.
[10] N. Maki, R. Shinkuma, T. Mori, N. Kamiyama, and R. Kawahara, "A periodic combined-content distribution mechanism in peer-assisted content delivery networks," Proc. ITU kaleidoscope, Kyoto, April 2013.
[11] "Yahoo! Japan," http://www.yahoo.co.jp/
[12] "ABC News," http://abcnews.go.com/
[13] "REUTERS," http://www.reuters.com/news/video
[14] G. Szabo and B.A. Huberman, "Predicting the popularity of online content," Commum. ACM, vol.53, no.8, pp.80–88, 2010.
[15] Y.J. Kim, T.U. Choi, K.O. Jung, Y.K. Kang, S.H. Park, and K.-D. Chung, "Clustered multimedia NOD: Popularity-based article prefetching and placement," IEEE Symposium on Mass Storage Systems, pp.194–202, California, 1999.
[16] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," Proc. INFOCOM'99, vol.1, pp.126–134, New York, March 2002.
[17] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat, "MediSyn: a synthetic streaming media service workload generator," Proc. 13th NOSSDAV '03, pp.12–21, California, June 2003.
[18] F.T. Johnsen, T. Hafsoe, C. Griwordz, and P. Halvorsen, "Workload characterization for news-on-demand streaming services," Proc. IEEE IPCCC'07, New Orleans, pp.314–323, April 2007.
[19] B. Tan and L. Massoulie, "Optimal content placement for peer-to-per video-on-demand systems," Proc. IEEE INFOCOM'11, pp.694–702, Shanghai, April 2011.
[20] G. Gao, R. Li, K. Wen, and X. Gu, "Proactive replication for rate objects in unstructured peer-to-peer networks," J. Network and Computer Applications, vol.35, no.1, pp.85–96, Jan. 2012.
[21] M.M. Amble, P. Parag, S. Shakkottai, and L. Ying, "Content-aware caching and traffic management in content distribution networks," Proc. INFOCOM'11, pp.2858–2866, Shanghai, April 2011.
[22] J.B. Chen, "Efficient content placement on multimedia cdn using fuzzy decision algorithm," Applied Mathematics & Information Science. Jan. 2012.
[23] N. Kamiyama, R. Kawahara, T. Mori, and H. Hasegawa, "Multicast pre-distribution in VoD service," Proc. IEEE CQR, pp.1–6, Naples, May 2011.
[24] R. Cuevas, M. Kryczka, A. Cuevas, S. Kaune, A. Guerrero, and

R. Rejaie, "Is content publishing in Bittorrent altruistic or prifit-driven?," ACM CoNEXT, 2010.
[25] S. Awiphan, A. Su, and J. Katto, "Providing service differentiation for video streaming in hybrid P2P overlay network," 19th International Packet Video Workshop, pp.13–18, Munich, May 2012.
[26] J.F. Paris and A. Amer, "Delayed chaining: a practical p2p solution for video-on-demand," Proc. IEEE ICCCN 2012, pp.1–5, Munich, Aug. 2012.
[27] "Amazon Instant Video," http://www.amazon.com
[28] "blinkbox," http://www.blinkbox.com

Naoya Maki received the B.E. degree in Electrical and Electronic Engineering from Kyoto University, Kyoto, Japan, in 2011. He is a master course student of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University. His current research interest is in traffic engineering for peer-assisted content delivery networks.

Ryoichi Shinkuma received the B.E., M.E., and Ph.D. degrees in Communications Engineering from Osaka University, Osaka, Japan, in 2000, 2001, and 2003, respectively. In 2003, he joined Communications and Computer Engineering, Graduate School of Informatics, Kyoto University as an Associate Professor. He was a Visiting Scholar at Wireless Information Network Laboratory (WINLAB), Rutgers, the State University of New Jersey, USA, from 2008 Fall to 2009 Fall. His research include network design and control criteria, particularly inspired by economic and social aspects. He received the Young Researchers' Award from IEICE in 2006 and the Young Scientist Award from Ericsson Japan in 2007, respectively. He is a member of IEEE.

Tatsuro Takahashi received the B.E., M.E. in Electrical Engineering from Kyoto University, Kyoto, Japan, in 1973 and 1975 respectively, and Dr. of Engineering in Information Science from Kyoto University in 1975 to 2000, making R&D on high speed networks and switching systems for circuit switching, packet switching, frame relaying, and ATM. Since July 1, 2000, he is a Professor, Communications and Computer Engineering, Graduate School of Informatics, Kyoto University. His current research interests include high-speed networking, photonic networks and mobile networks. Prof. Takahashi received the Achievement Award from IEICE in 1996, the Minister of Science and Technology Award in 1998, and the Distinguished Achievement and Contributions Award from IEICE in 2011. He was a Vice President of the ATM Forum from 1996 to 1997, and the Chairman of the Network Systems (NS) Technical Group in the Communications Society of IEICE from 2001 to 2002. Prof. Takahashi is an IEEE Fellow.