Optimizing Cache Location and Route on CDN Using Model Predictive Control

Noriaki Kamiyama[1,2], Yousuke Takahashi[2], Keisuke Ishibashi[2], Kohei Shiomoto[2],
Tatsuya Otoshi[1], Yuichi Ohsita[1], and Masayuki Murata[1]
[1]Osaka University
[2]NTT Network Technology Laboratories
E-mail: kamiyama.noriaki@ist.osaka-u.ac.jp

***Abstract*—In content delivery services, cache-server selection and route control are independently operated by content delivery network (CDN) providers and Internet service providers (ISPs), respectively. However, the number of ISPs providing CDN service has been increasing, and they can optimize the cache-server and delivery-route selection simultaneously. In this paper, we investigate the effect of jointly controlling these two operations. To continuously maintain a desirable state over a long time span in an environment in which estimating future demand is not easy, we should use an optimization method with which we can repeat the optimization procedure continuously. Therefore, we propose a method of optimizing the cache-server selection using the model predictive control (MPC), which has been widely used in system control. We also propose a method of simultaneously optimizing both the cache-server and delivery-route selection using the MPC. Through numerical evaluation using two actual network topologies of Tier-1 ISPs and the access log data of VoD services, we confirm that we can achieve almost the same effect by optimizing only cache-server selection using the MPC compared with optimizing both operations simultaneously. We also confirm that the number of failed requests that are not accommodated in the system can be reduced by about 1/30 with the proposed methods using the MPC.**

I. Introduction

Internet routers switch packets in accordance with the destination IP address without keeping flow states, and packets are transmitted on the route with the minimum total value for the static weights of links. The sending hosts autonomously determine the flow rates without call admission control, so link congestion is inevitable. Therefore, to maintain adequate quality, Internet service providers (ISPs) need to design and manage their networks so that the loads on all links are kept below a certain level. However, the link loads change due to variations in the traffic demand matrix and to route fluctuations caused by failures. This led to the development of various traffic engineering (TE) methods that balance the link loads by dynamically configuring the packet routes [4][11][13][15][16][19][21][24].

In recent years, a large percentage of Internet traffic has been dominated by HTTP traffic. For example, an analysis of traffic measured on a backbone link between Japan and the USA from 2006 to 2008 indicated that about 60 percent of the traffic consisted of HTTP packets [5]. Moreover, user generated content (UGC) traffic, e.g., YouTube, and rich content of VoD services has dramatically increased, and it has been reported that the amount of traffic generated by these content delivery services account for more than 80% of the total traffic on the Internet [10]. Content delivery networks (CDNs) that use a number of cache servers deployed in multiple networks have been widely used to efficiently deliver various types of content [26][28][34]. Currently, 74% of the 1,000 most frequently accessed websites use CDNs [28].

978-1-4799-0913-1/14/$31.00 © 2014 IEEE

CDN providers provide cache servers in a large number of ISP networks and redirect a user′s request to the cache server that is closest to the user using Domain Name System (DNS). Internet service providers are reluctant to inform their network status, e.g., link utilization and network topology, to CDN providers, so CDN providers normally select cache servers without considering the network status. However, cache-server selection determines the location of the source node of delivery flows, which will strongly affect the traffic demand matrix among network nodes, so the server selection of CDN providers affects the TE operated by ISPs. Because of this interaction between CDN providers and ISPs, it is difficult to maximize the effect of TE in content delivery services.

Although CDNs are typically operated by CDN providers, e.g., Akamai, the number of CDNs operated by large-scale content providers, such as Google, and by tier-1 ISPs, such as AT&T, has been increasing recently [25]. Collaboratively controlling server selection and routes by sharing various information between CDN providers and ISPs has also been investigated [14][30]. Therefore, in these CDNs, the cache-sever and delivery-route selection can be jointly optimized by a single provider. Moreover, various CONs (content oriented networks), e.g., ICN (information-centric networking) and CCN (content-centric networking), in which packets are routed based on the content name instead of destination network address have been proposed [1][9][20]. In CONs, ISPs operate cache memories provided at routers by themselves, so it is realistic to optimize the cache-server and delivery-route selection simultaneously.

The demand for each content item continues to change dramatically, so we need to adaptively control server selection and route according to the demand dynamics to optimize them simultaneously. Therefore, various methods of estimating the future demand of each content item have been proposed

[2][6][8][17]. However, we cannot completely avoid the estimation error in predicting the future demand of content, so it is effective to control server selection and routes by taking into account estimation error.

The model predictive control (MPC) has gained wide attention as an adaptive control method that is robust against estimation error in control engineering [31]. Similar to conventional control methods, the MPC also optimizes the system parameters so that the output of the system approaches the target value. However, unlike conventional control methods, the MPC optimizes input by taking into account the outputs in multiple succeeding times slots, i.e., *predictive horizon*, simultaneously. The MPC also takes into account minimizing the change in inputs between two adjacent time slots to improve the robustness against estimation error and suppress the impact on the system. By repeating this optimization process at each time slot, the MPC can adaptively optimize the system by modifying the input value based on the feedback of the difference between the output and target value. By using the MPC, we can expect to keep networks stable even if estimation error is not totally avoided. We have applied the MPC to control the routes of packets [27].

In this paper, we first propose a method of adaptively optimizing cache-server selection based on the MPC. Next, we apply the MPC to set the cache-server and delivery-route simultaneously. The contributions of this manuscript can be summarized as follows.

- We propose a method of optimizing cache-server selection using the MPC in content delivery services.
- We also propose a method of simultaneously optimizing cache-server and delivery-route selection using the MPC in content delivery services.
- Through computer simulation using the network topologies of commercial Tier-1 ISPs and the access logs of VoD services, we argue that the number of failed requests that are not accommodated in the system can be reduced by about 1/30 with the proposed methods using the MPC.

After briefly reviewing related work in Section II, we summarize the conditions assumed in this paper in Section III. In Section IV, we describe the two proposed methods of optimizing cache-server and deliver-route selection using the MPC. In Section V, we explain numerical results and conclude this manuscript in Section VI.

II. Related Work

Various methods have been proposed for controlling the routes of packets. For example, Fortz et al. proposed a method of optimally designing the link weights used in the open shortest path first (OSPF) routing protocol of the Internet to minimize the total link cost, which is a function of the traffic load on links [15][16]. Benson et al. proposed to set the routes in short time intervals based on the estimation of traffic demand in a short period [4], Danna et al. formalized the optimization problem of setting the routes as a linear programming problem by considering the tradeoff between the fairness of bandwidth allocated to users and utilization of network resources [11], and Kandula et al. proposed to adaptively set the routes of packets based on the measured throughput [24]. Although these TE methods set the routes in a coarse granularity, i.e., the destination IP address, there are TE methods that control the routes in finer granularity. For example, Elwalid et al. proposed to set the routes of label switched paths (LSPs) by minimizing the total link cost in MPLS networks [13], and Hong et al. and Jain et al. proposed to switch the routes with fine granularity in a short time scale using the SDN in inter-datacenter networks [19][21]. He et al. argued that end users control the routes by using an online optimization approach based on the measured network states [18], and Sharma et al. analyzed the effect of various TEs on user-perceived qualities [33].

There have also been various method proposed for selecting cache servers for each content-delivery request. For example, Poese et al. proposed a method of enabling CDN providers to optimize cache-server selection by using the information of network states provided by ISPs [30]. Instead of controlling cache-server selection, Applegate et al. proposed to design a location where each content is cached for minimizing the total amount of traffic in the networks with the constraints of the upper bounds of cache-storage capacity and link bandwidth [3]. Tang et al. proposed a method of designing the content location for minimizing the replication cost required to modify the content location by considering the constraints on user qualities [35].

As these examples suggest, there are various methods for ISPs to control the routes of packets or for CDN providers to control cache-server selection independently. However, these methods do not take into account the interaction between the route control of ISPs and cache-server selection of CDN providers. The route selection of ISPs affects cache-server selection of CDN providers, and cache-server selection of CDN providers affects the route control of ISPs, so we need to consider the interaction between these two controls independently operated by ISPs and CDN providers when optimizing these controls. DiPalantino et al. investigated the strategies between ISPs and CDN providers using game theory and analyzed the resulting equilibrium states [12]. Poese et al. found that the addresses that the DNS server answered in content delivery services were versatile by inspecting the captured packet data, and we can expect to reduce the response time if ISPs select the cache servers [29]. Frank et al. proposed a method of exchanging the information between ISPs and CDN providers when these two players collaboratively control cache-server selection [14]. However, none of these studies were focused on the simultaneous control of cache-server selection and routes by ISPs.

Jiang et al. investigated the effect of exchanging the information between ISPs and CDN providers when these two players controlled the routes and cache-sever selection independently [22]. Moreover, Rossini et al. proposed a method of controlling the cache location and forwarding toute of each content simultaneously [32]. However, they optimized at a single time instance, and it is difficult to apply their methods

to actual networks in which the traffic demand continues to change. To achieve desirable states in multiple time periods in an environment in which estimating future traffic demand is difficult, we need methods that enable ISPs to repeat the optimization procedure continuously over multiple time instances.

III. Assumptions

In this section, we describe the various conditions assumed in this paper. Table I summarizes the definitions of the main notations.

TABLE I
Definition of Symbols

Notation	Definition
T	Length of time slot (TS)
N	Number of nodes
E	Number of links
$C_{L,e}$	Transmission capacity of link e
$\boldsymbol{R}_{i,j}$	Set of routes from node i to node j
M	Number of content items
$d_{u,m,t}$	Request count for content m from user u in TS t
$\hat{d}_{u,m,t}$	Estimate of $d_{u,m,t}$
o_m	Node where original content m exists
B_s	Storage capacity of cache server s
$C_{C,s}$	Processing capacity of cache server s
$\boldsymbol{S}_{m,t}$	Cache set storing content m at beginning of TS t
$\boldsymbol{\Phi}_{u,m}$	Set of delivery routes of content m for user u
$\phi_{u,m}(s,r)$	Route r of content m from cache s to user u
$p_{u,m,t}(s)$	Probability of selecting cache server s when sending content m to user u in TS t
$\acute{p}_{s,u,t}(r)$	Probability of selecting route r when sending content from server s to user u in TS t
$\tilde{p}_{u,m,t}(s,r)$	Probability of selecting $\phi_{u,m}(s,r)$ when sending content m to user u
$I_{s,u,t}(r,e)$	Binary variable taking unity when route r from node s to node u takes link e in TS t
$f_{e,t}$	Number of delivery flows taking link e in TS t
$\eta_{e,t}$	Delivery-flow count exceeding capacity of link e
$g_{s,t}$	Number of delivery flows served by cache server s in TS t
$\zeta_{s,t}$	Delivery-flow count exceeding capacity of cache server s
$h_{s,u,r,t}$	Hop length of route r from nodes s to u in TS t
H	Predictive horizon length
J_s	Optimization function related to congestion of cache servers
J_e	Optimization function related to link congestion
J_h	Optimization function related to hop length of delivery flows
$J_{v,p}$, $J_{v,\tilde{p}}$, $J_{v,\acute{p}}$	Optimization function related to change of $p_{u,m,t}(s)$, $\tilde{p}_{u,m,t}(s)$, $\acute{p}_{u,m,t}(s)$
w_e, w_h, $w_{v,p}$, $w_{v,\tilde{p}}$, $w_{v,\acute{p}}$	Weights of J_e, J_h against J_s Weights of $J_{v,p}$, $J_{v,\tilde{p}}$, $J_{v,\acute{p}}$ against J_s

A. Mechanism of CDN

For each user request, CDN selects the cache server from which the requested content is delivered to users. We assume the procedure selecting the delivery server used in Akamai CDN [26]. To resolve the IP address of a content server from the content name, a user terminal first sends a query message to the local DNS server, and the local DNS server replies with the IP address of the Akamai DNS server to the requesting user terminal, as shown with arrow (1) in Fig. 1. Next, as shown with arrow (2), the user terminal sends the DNS query message to the Akamai DNS server, which selects one cache server and returns the IP address of the selected cache server to the user terminal. Although the criteria for selecting cache servers is not open to the public, it was reported that the cache server, from which the measured response time is minimum or located closest to the user terminal, is selected in many cases [34].

As shown with arrow (3) in Fig. 1, the user terminal requests the resolved cache server to deliver the content, and this cache server sends the content when it has already cached the requested content, i.e., *cache hit*. When the requested content is not cached at this cache server, i.e., *cache miss*, this cache server obtains the content from the origin server, as shown with arrow (4) in Fig. 1, and it sends the obtained content to the user terminal. The resolved cache server stores the obtained content in its local storage after removing some of the previously cached content items based on a cache replacement policy. In many cases, the least recently used (LRU) policy of removing content with the longest elapsed time from the final request is used as the cache replacement policy.

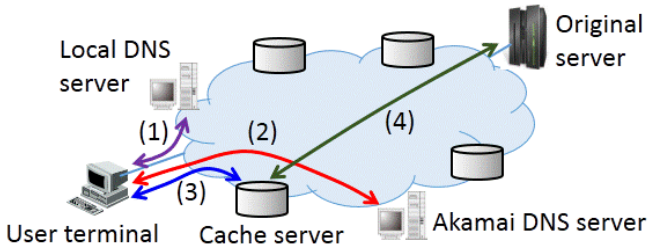


Fig. 1. Mechanism of CDN

B. Networks

We assume a network constructed and managed by a single ISP with N nodes and E links, and the entire network is operated based on time slots with fixed length of T. The transmission capacity of link e, i.e., the total number of content items that can be transmitted on link e within each time slot, is given by $C_{L,e}$. A set of candidate routes, which are used to deliver content from node i to node j, $\boldsymbol{R}_{i,j}$, is statically given, and the content-delivery route is determined by selecting any route r of $r \in \boldsymbol{R}_{i,j}$ when delivering content from node i to node j. The destination node of content delivery to user u, i.e., users accommodated into node u, is node u.

C. Content

We assume that M content items with identical size are provided throughout the entire network. Let $d_{u,m,t}$ denote the number of requests for content m generated from user u in time slot t. Origin servers are provided at all the N nodes, and the original of each content item is provided at just a single node. The location of the original content is fixed, and let o_m denote the node having the original of content m. We describe the origin server provided at note n as *origin server* n. We also assume that the throughput of delivery flows is identical and stable.

D. Cache Server

Cache servers are provided at all N nodes, and we denote the cache server at node n as *cache server* n, and *server* n stands for both origin server n and cache server n. The storage capacity of cache server n, i.e., the maximum number of content items that can be cached at cache server n, is given by B_s. The processing capacity for delivering content at server n is finite, so we define $C_{C,s}$ as the maximum number of concurrent delivery sessions that can be supported by server s. Only when cache server s, which is selected for delivering content m to user u, has content m, i.e., cache hit, and the unused capacities of cache server s as well as all the links on the selected route r are sufficient to accommodate the new delivery session, content m is delivered to user u from cache server s, and delivery flow is generated from node s to node u.

In the other cases, i.e., cache miss or when exceeding processing capacity of transmission capacity of cache server s or links of route r, content m is delivered from origin server o_m to node u via node s, as mentioned in Section III-A, if the available processing capacity of o_m and the transmission capacity of links on the route is sufficient to accommodate the new session. In this case, two delivery flows are used: one is from node o_m to node s and the other is from node s to node u, and we assume the minimum hop route for the former delivery flow. Moreover, cache server s stores content m at its local storage after removing content based on the LRU cache replacement policy. If the processing capacity of origin server o_m or the transmission capacity of links on the routes from node o_m to node u via node s is not enough, the request is rejected, and we assume that users of failed requests request the same content again in the next time slot. The content that each cache server stores dynamically changes, and let $\boldsymbol{S}_{m,t}$ denote the set of cache servers that have content m at the beginning of time slot t.

IV. Cache Server and Route Selection Using MPC

A. Model Predictive Control (MPC)

The model predictive control (MPC) as an adaptive control method that is robust against estimation error has gathered wide attention in the control engineering [31]. Like conventional control methods, the MPC also optimizes the system parameters so that the output of the system approaches the target value. However, unlike the conventional methods, the MPC optimizes the input by considering the outputs in multiple succeeding times slots, i.e., *predictive horizon*, simultaneously. Let y_k and Y_k denote the output of the system and its target value in time slot t, respectively, and x_t be the input of the system in time slot t. At the ending of time slots t, the MPC derives the series of inputs in the predictive horizon, i.e., the most recent H time slots, $x_{t+1}, x_{t+2}, \cdots, x_{t+H}$, so that $J_1 = \sum_{k=t+1}^{t+H} \parallel y_k - Y_l \parallel^2$, i.e., the sum of the deviations of y_k from its target value Y_k over these H time slots, is minimized, where $\parallel x \parallel$ represents the Euclidean norm of x.

To derive the inputs in the predictive horizon, the MPC requires the estimates of output y_k when giving input x_k in each time slot k in the predictive horizon. The time evolution of outputs y_k against inputs x_k considering the system state in time slot k, z_k, are represented using the state space model:

$$z_{k+1} = f(k,\ z_k,\ x_k), \tag{1}$$

$$y_k = g(k,\ z_k,\ x_k), \tag{2}$$

where f and g are the functions defining the relationship among x_k, y_k, and z_k. Completely avoiding the estimation error of y_k is infeasible, and the estimation error increases as k increases in many cases, so the MPC actually inputs only x_{t+1} to the system among the derived H inputs, $x_{t+1}, \cdots, x_{t+H}$, in the predictive horizon $[t+1,\ t+H]$. By repeating this optimization procedure at each time slot, the MPC rectifies the inputs x_k based on the feedback obtained from the measured output y_k.

If the input x_k is set to a value largely different to the previous input x_{k-1} as a result of being affected by estimation error, the system might be unstable. To sustain a stable state of the system by avoiding excess change in x_k, the MPC attempts to reduce $\parallel \Delta x_k \parallel$, the amount of change between x_k and x_{k-1}, where Δx_k is defined as $\Delta x_k = x_k - x_{k-1}$. To achieve this goal, the MPC attempts to simultaneously decrease both J_1 and J_2, the total amount of change in x_k in the predictive horizon $[t+1,\ t+H]$, where J_2 is given by $J_2 = \sum_{k=t+1}^{t+H} \parallel \Delta x_k \parallel$. In other words, the MPC derives the inputs x_{t+1}, x_{t+2}, $\cdots$, x_{t+H} by minimizing $J_1 + wJ_2$, where w is a parameter determining the balance between J_1 and J_2.

We illustrate the optimization process of the MPC in Fig. 2. The system state in time slot $t+1$, z_{t+1}, is determined by the system state z_t and input x_t in the previous time slot t, and the output in time slots $t+1$, y_{t+1}, is determined by z_{t+1} and x_{t+1}. Therefore, when the inputs x_k in the predictive horizon, $t+1 \leq k \leq t+H$, are given, the output y_k in each time slot k of the predictive horizon can be recursively derived from $k = t+1$, and the optimization target function $J_1 + wJ_2$ is obtained. The MPC derives x_k of $t+1 \leq k \leq t+H$ by solving the nonlinear optimization problem of minimizing this optimization target function, and the obtained x_{t+1} is inputted to the system.

B. Optimization of Server Selection Using MPC

In this section, we discuss the application of the MPC to the optimization problem of cache-server selection. Although the system state z_t is $d_{u,m,t}$, the demand for content m

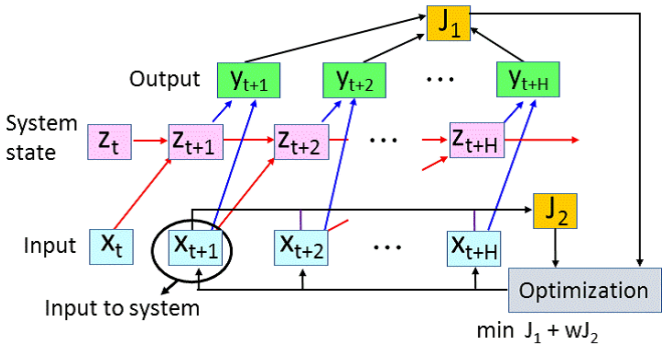


Fig. 2. Optimization flow of model predictive control

generated from user u in time slot t, which is given by the external environment, we use its estimation $\hat{d}_{u,m,k}$ instead of $d_{u,m,t}$ because $d_{u,m,t}$ of $k \geq t$ is unknown at the beginning of time slot t. If we deterministically select just one cache server used for requests of content m from user u, the MPC optimization problem becomes an integer programming problem, and deriving the inputs x_t within a practical timeframe is difficult. Therefore, we consider selecting cache servers probabilistically. In other words, we select cache server s with probability $p_{u,m,t}(s)$ from the candidate set $\boldsymbol{S}_{m,t}$ for each request of content m from user u in time slot t. The input x_t of the MPC is $p_{u,m,t}(s)$, and we can derive $p_{u,m,t}(s)$ within a short timeframe at the beginning of each time slot because the MPC optimization problem becomes a nonlinear programming problem of minimizing the objective function with the constraint that $\sum_{s\in\boldsymbol{S}_{m,t}} p_{u,m,t}(s) = 1$.

We can consider balancing the load of cache servers, reducing the number of failed requests due to congestion of cache servers, and decreasing the hop length of delivery flows as the optimization targets of cache-server selection in content delivery services. Let $\zeta_{s,t}$ denote the number of failed requests at server s in time slot t, which is given by $\zeta_{s,t-1} = [g_{s,t-1} - C_{C,s}]^+$, where $[x]^+$ is x when $x > 0$ and zero when $x \leq 0$. Using $\zeta_{s,t}$, we obtain $g_{s,t}$, the total number of delivery flows provided by server s in time slot t, as

$$g_{s,t} = \sum_{m=1}^{M}\sum_{u=1}^{N}\sum_{r\in\boldsymbol{R}_{s,u}} \hat{d}_{u,m,t} p_{u,m,t}(s) + \zeta_{s,t-1}. \quad (3)$$

We define the optimization function related to the congestion of cache servers J_s as

$$J_s = \sum_{k=t+1}^{t+H}\sum_{s=1}^{N} \zeta_{s,k}. \quad (4)$$

Moreover, to reduce the average hop length of delivery flows, we define the second optimization function J_h as

$$J_h = \sum_{k=t+1}^{t+H}\sum_{m=1}^{M}\sum_{u=1}^{N}\sum_{s\in\boldsymbol{S}_{m,k}} h_{s,u,k}\hat{d}_{u,m,k} p_{u,m,k}(s), \quad (5)$$

where $h_{s,u,k}$ is the hop length on the delivery route from node s to node u, and we assume the minimum-hop route for the delivery flows. Moreover, to reduce the amount of change of $p_{u,m,t}(s)$, we define the third optimization function $J_{v,p}$ as

$$J_{v,p} = \sum_{k=t+1}^{t+H}\sum_{m=1}^{M}\sum_{u=1}^{N}\sum_{s\in\boldsymbol{S}_{m,k}} \left| p_{u,m,k}(s) - p_{u,m,k-1}(s) \right|. \quad (6)$$

To consider these three optimization functions (J_s, J_h, and $J_{v,p}$), we define the optimization function as $J_s + w_h J_h + w_{v,p} J_{v,p}$, where w_h and $w_{v,p}$ are the weights of J_h and $J_{v,p}$ against J_s. The MPC derives $p_{u,m,k}(s)$ of each k in the range $t+1 \leq k \leq t+H$ at the end of time slot t by solving the following nonlinear programming problem:

$$\begin{aligned}
\min \quad & J_s + w_h J_h + w_{v,p} J_{v,p}, && (7)\\
s.t. \quad & \zeta_{s,t} = [g_{s,t-1} - C_{C,s}]^+, \ \forall s, && (8)\\
& g_{s,t} = \sum_{m=1}^{M}\sum_{u=1}^{N}\sum_{r\in\boldsymbol{R}_{s,u}} \hat{d}_{u,m,t} p_{u,m,t}(s,r) \\
& \qquad + \zeta_{s,t-1}, && (9)\\
& 0 \leq p_{u,m,t}(s) \leq 1, \ \forall u, \forall m, \forall s, && (10)\\
& \sum_{s\in\boldsymbol{S}_{m,t}} p_{u,m,t}(s) = 1, \ \forall u, \forall m. && (11)
\end{aligned}$$

C. Simultaneous Optimization of Cache-Server and Delivery-Route Selection Using MPC

The content-delivery flow generated for each request is uniquely determined by selecting the delivery server and delivery route, and we call a set of delivery servers and routes a *delivery path*. In this section, we consider optimizing the delivery path used for delivering content m to user u using the MPC. We define $\boldsymbol{\Phi}_{u,m,t}$ as the set of delivery paths that can be used for delivering content m to user u in time slot t and denote $\phi_{n,m,t}(s,r)$ as the delivery path with source node s ($s \in \boldsymbol{S}_{m,t}$) and delivery route r ($r \in \boldsymbol{R}_{s,u}$) among the delivery paths included in $\boldsymbol{\Phi}_{u,m,t}$. Figure 3 illustrates examples of delivery paths for delivering content m to user A or B when content m is cached at cache server X and Y. When delivering content m to user A, there is one delivery path $\phi_{A,m,t}(X,1)$ from cache server X and two delivery paths $\phi_{A,m,t}(Y,1)$ and $\phi_{A,m,t}(Y,2)$ from cache server Y, so there is a total of three delivery paths when delivering content m to user A. Similarly, five delivery paths exist to user B, i.e., three delivery paths from cache server A and two delivery paths from cache server B.

We also consider selecting delivery paths probabilistically to derive the input x_t of the MPC within a short timeframe. In other words, we select delivery path $\phi_{u,m,t}(s,r)$ with the probability of $\tilde{p}_{u,m,t}(s,r)$ from the candidate set $\boldsymbol{\Phi}_{u,m,t}$ when user u requests content m in time slot t. The input x_t of the MPC is $\tilde{p}_{u,m,t}(s,r)$, and we can derive $\tilde{p}_{u,m,t}(s,r)$ within a short timeframe at the beginning of time slot t by solving the nonlinear programming problem of minimizing the object function with the constraint that $\sum_{s\in\boldsymbol{S}_{m,t}}\sum_{r\in\boldsymbol{R}_{s,u}} \tilde{p}_{u,m,t}(s,r) = 1$.

For simultaneously optimizing cache-server and delivery-route selection, we can consider reducing the amount of traffic

within the network and the number of failed requests due to congestion at cache servers or network links. To reduce the amount of traffic within the network, decreasing the hop length of delivery flows is effective, and keeping the load of cache servers and links below their capacities is required to reduce the number of failed requests due to congestion at cache servers or network links. Controlling the delivery routes affects the link load and hop length of delivery flows, and the selection of cache servers affects the cache-server load in addition to the link load and flow-hop length. Therefore, we need to simultaneously consider reducing these three metrics in the predictive horizon $[t+1,\ t+H]$ when designing the selection probability of delivery paths.

By using $I_{s,u,t}(r,e)$, a binary variable taking unity when candidate route r from node s to node u in time slot t takes link e, and $\eta_{e,t}$, the number of requests that are blocked due to overload of link e in time slot t, we obtain $f_{e,t}$, the total number of delivery flows transmitted on link e in time slot t, as

$$f_{e,t} = \sum_{m=1}^{M}\sum_{u=1}^{N}\sum_{s\in\boldsymbol{S}_{m,t}}\sum_{r\in\boldsymbol{R}_{s,u}} I_{s,u,t}(r,e)\hat{d}_{u,m,t}\tilde{p}_{u,m,t}(s,r)+\eta_{e,t-1}, \quad (12)$$

where $\eta_{e,t-1}$ is given by $\eta_{e,t-1} = [f_{e,t-1} - C_{L,e}]^+$. We define J_e as the optimization target function related to the link congestion, and it is given by

$$J_e = \sum_{k=t+1}^{t+H}\sum_{e=1}^{E}\eta_{e,k}. \quad (13)$$

To reduce the amount of change in $\tilde{p}_{u,m,t}$, we also define the second optimization target function $J_{v,\tilde{p}}$ as

$$J_{v,\tilde{p}} = \sum_{k=t+1}^{t+H}\sum_{m=1}^{M}\sum_{u=1}^{N}\sum_{s\in\boldsymbol{S}_{m,k}}\sum_{r\in\boldsymbol{R}_{s,u}} \left|\tilde{p}_{u,m,k}(s,r)-\tilde{p}_{u,m,k-1}(s,r)\right|. \quad (14)$$

We consider J_e and $J_{v,\tilde{p}}$ in addition to J_s and J_h defined in Section IV-B as the optimization targets, and we define the optimization target function as $J_s + w_e J_e + w_h J_h + w_{v,\tilde{p}} J_{v,\tilde{p}}$, where w_e and $w_{v,\tilde{p}}$ are the weights of J_e and $J_{v,\tilde{p}}$ against J_s, respectively. The MPC derives $\tilde{p}_{u,m,k}(s,r)$ for each k of $t+1 \le k \le t+H$ by solving the following nonlinear programming problem:

$$\min \quad J_s + w_e J_e + w_h J_h + w_{v,\tilde{p}} J_{v,\tilde{p}} \quad (15)$$

$$s.t. \quad \eta_{e,t} = [f_{e,t-1} - C_{L,e}]^+,\ \forall e, \quad (16)$$

$$\zeta_{s,t} = [g_{s,t-1} - C_{C,s}]^+,\ \forall s, \quad (17)$$

$$f_{e,t} = \sum_{m=1}^{M}\sum_{u=1}^{N}\sum_{s\in\boldsymbol{S}_{m,t}}\sum_{r\in\boldsymbol{R}_{s,u}} I_{s,u,t}(r,e)\hat{d}_{u,m,t}\tilde{p}_{u,m,t}(s,r)+\eta_{e,t-1}, \quad (18)$$

$$g_{s,t} = \sum_{m=1}^{M}\sum_{u=1}^{N}\sum_{r\in\boldsymbol{R}_{s,u}} \hat{d}_{u,m,t}\tilde{p}_{u,m,t}(s,r) + \zeta_{s,t-1}, \quad (19)$$

$$0 \le \tilde{p}_{u,m,t}(s,r) \le 1,\ \forall u,\forall m,\forall s,\forall r, \quad (20)$$

$$\sum_{s\in\boldsymbol{S}_{m,t}}\sum_{r\in\boldsymbol{R}_{s,u}} \tilde{p}_{u,m,t}(s,r) = 1,\ \forall u,\forall m. \quad (21)$$

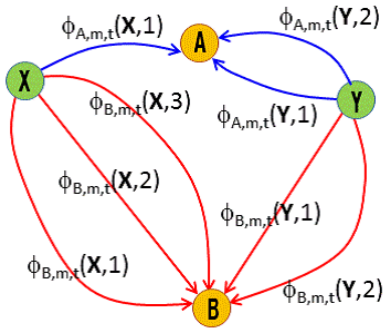


Fig. 3. Example of delivery paths of content m to user A or B

V. Numerical Evaluation

A. Comparison Methods

We compared three cache-server-selection methods, *Local*, *Cooperate*, and *MPC*, for requesting content m from user u. In Local, only the local cache, i.e., cache server u, is always selected, whereas the cache server with the shortest hop length to node u among all the cache servers having content m is selected in Cooperate. In MPC, cache servers are probabilistically selected according to the probabilities designed by the MPC. On the other hand, we compared the two selection methods of delivery routes, *Shortest* and *MPC*. In Shortest, the minimum-hop routes are always selected between any pair of nodes, whereas in MPC, the delivery routes are probabilistically selected according to the probabilities designed by the MPC. Therefore, we compared a total of six methods: Local-Shortest, Local-MPC, Cooperate-Shortest, Cooperate-MPC, MPC-Shortest, and MPC-MPC. The method we proposed in Section IV-B corresponds to MPC-Shortest, and that we proposed in Section IV-C corresponds to MPC-MPC.

Local-MPC and Cooperate-MPC correspond to the case in which ISPs without operating CDNs control the routes of packets using the MPC, and $\acute{p}_{s,u,t}(r)$, the probability that route r of $r \in \boldsymbol{R}_{s,u}$, is selected as the route from server s to user u. ISPs cannot know the location where each content item is cached and the policy of CDN providers in selecting the cache servers, so it is difficult for ISPs to estimate the traffic demand between each pair of nodes. However, it is expected that the local cache servers that are closest to the requesting users are selected in many cases, and traffic is not generated in the case of cache hits. Therefore, $\acute{p}_{s,u,t}(r)$ is designed using the MPC by assuming that content m is delivered from its origin server o_m to user u. Defining the optimization function $J_{v,\acute{p}}$ as the amount of change in $\acute{p}_{s,u,t}(r)$, similarly to the definition of $J_{v,p}$ and $J_{v,\tilde{p}}$, the optimization target function is given by $w_e J_e + w_h J_h + w_{v,\acute{p}} J_{v,\acute{p}}$, where $w_{v,\acute{p}}$ is the weight of $J_{v,\acute{p}}$ against J_s.

B. Evaluation Conditions

We used the topologies of the two commercial ISP backbone networks in USA, CAIS, and Verio [7], and Fig. 4 shows the topologies of these two networks. The number of nodes, N, was 37 and 35, the number of links, E, was 88 and 152, and the average node degree was 2.38 and 4.11 in CAIS and Verio, respectively. Verio had a larger average node degree compared with CAIS, and there were hub nodes with a large degree in Verio. On the other hand, CAIS had a smaller average node degree and no hub nodes. In Verio, the average hop distance between each pair of nodes was small because of the existence of hub nodes, so the average hop distance between nodes was 2.51 in Verio, whereas it was 5.05 in CAIS.

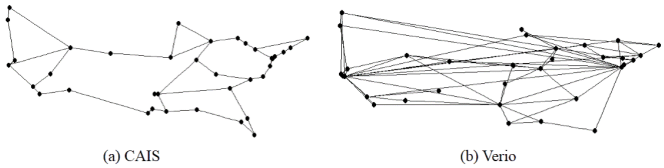
We generated content requests according to the access log data of the PowerInfo VoD system, which was a commercial VoD service operated by China Telecom [36]. This log data consists of 20,921,657 requests for seven months from June 2004 to December 2004, and we used the access logs for 10 days from the 162-nd day to the 171-st day. At the time of each request of log data in these 10 days, we generated a request at a node randomly selected with the probability proportional to the node population. We started the computer simulation with the initial state in which no content items were cached at all the N nodes, and we evaluated the six methods in the last seven days after removing the data obtained in the first three days as the warm-up period.

We assumed that cache servers are provided at all the N nodes. The access log data in the evaluation period contained 7,500 content items, and we set the storage capacity of each cache server as $B_s = 7,500/N$. We set the length of time slots as $T = 30$ minutes. The maximum number of requests in one time slot was 5,700, and we set $C_{C,s}$, the processing capacity of each server, and $C_{L,e}$, the transmission capacity of each link, as $C_{C,s} = 5,700 \times 2/N$ and $C_{L,e} = 5,700 \times 4/N$. Before starting the computer simulation, we assigned o_m, the origin server of content m, to the node randomly selected with the probability proportional to the node population, and the location of origin servers was fixed during the computer simulation.

We set the length of predictive horizon as $H = 2$ and the weights of each optimization target function as $w_e = 0.5$ and $w_h = w_{v,p} = w_{v,\tilde{p}} = w_{v,\acute{p}} = 0.001$. We used the actual demand of content, $d_{u,m,t}$, as its estimation $\hat{d}_{u,m,t}$. In MPC-Shortest, (Local or Cooperate)-MPC, and MPC-MPC, we randomly selected the cache server, delivery route, and delivery path according to $p_{u,m,t}(s)$, $\tilde{p}_{u,m,t}(s,r)$, and $\acute{p}_{u,m,t}(r)$ for each request of content m from user u in time slot t, respectively. We used the three k-shortest paths i.e., the three minimum-hop routes, as $\boldsymbol{R}_{i,j}$, the candidate set of routes from node i to node j.

C. Numerical Results

We compared the average number of failed requests that were not accepted due to congestion at the delivery server or network links in each time slot and the average hop length of delivery flows among the six methods described in Section V-A. Figure 5 plots the average number of failed requests in each time slot of the six methods. As mentioned in Section V-B, we did not consider the results in the first three days as the warm-up period, and we evaluated the results in the succeeding seven days, so the range of the x-axis is from the 145-th time slot to the 480-th time slot. We observed a variation pattern with a cycle of 48 time slots, i.e., 24 hours, with all the six methods. Tables II and III show the average number of failed requests of each of the six methods in CAIS and Verio. We confirmed that the effect of the server-selection method on the failed-request count was much larger than that of the route-selection method, and the cache-server-selection method was a major factor in determining the failed-request count.



(a) CAIS (b) Verio

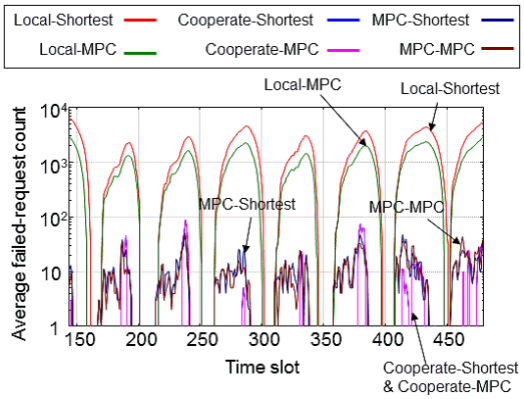
Fig. 4. Network topology of CAIS and Verio

When the cache-sever-selection method was Local, the candidate of cache servers for each request of user u was limited to cache server u, so the failed-request count was one or two orders larger than those when using Cooperate or MPC as the cache-server-selection method. When Cooperate was used as the cache-server-selection method, the cache server with the shortest hop distance to the requesting user was always selected among all the cache servers having the requested content at the time of user request. When MPC was used as the cache-server-selection method, on the other hand, the selection probabilities of cache servers were designed based on the cache deployment at the beginning of each time slot, and the actual content deployment at the time of user request was not reflected in the cache-server selection. Although the processing load required in the cache-server selection of MPC was much smaller than that required in the cache-server selection of Cooperate, it was anticipated that the failed-request count would increase with MPC compared with Cooperate.

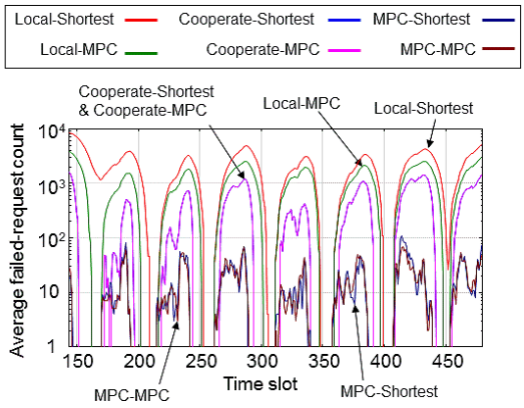
However, the failed-request count of MPC cache-server selection was just slightly larger than that of Cooperate cache-server selection in CAIS. Moreover, in Verio, the failed-request count of MPC cache-server selection was much smaller than that of Cooperate cache-server selection, and we confirmed that the number of failed requests that were not accommodated in the network can be reduced by about 1/30 by using MPC cache-server selection. Verio had several hub nodes, and the hop distance to other nodes from the hub nodes tended to be small, so the cache servers located at hub nodes were selected for many requests, and the cache servers at hub nodes were overloaded. Therefore, in hub & spoke type networks, in which hub nodes existed, the effect of using MPC as the cache-server selection was especially high because many cache servers were

probabilistically selected, and the diversity of selected cache servers was high when using the MPC.

Figure 6 plots the average hop length of delivery flows in each time slot of the six methods. We also observed a variation pattern with a cycle of 48 time slots with all the six methods. Tables IV and V summarize the average hop length of delivery flows in each of the six methods on CAIS and Verio, respectively. We also confirmed that the effect of route selection on the results was negligible, and cache-server selection was the dominant factor in determining the flow hop length. Although the average hop length of flows using the Cooperate cache-server selection was the smallest in both networks, MPC cache-server selection also achieved similar results with Cooperate cache-server selection.



(a) CAIS network



(b) Verio network

Fig. 5. Average number of failed requests in each time slot

TABLE II
AVERAGE NUMBER OF FAILED REQUESTS IN CAIS NETWORK

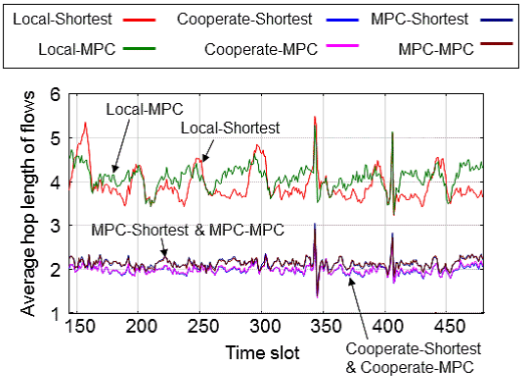
Route selection	Cache-server selection		
	Local	Cooperate	MPC
Shortest	1548.1	3.69	7.59
MPC	800.3	3.69	7.24

VI. CONCLUSION

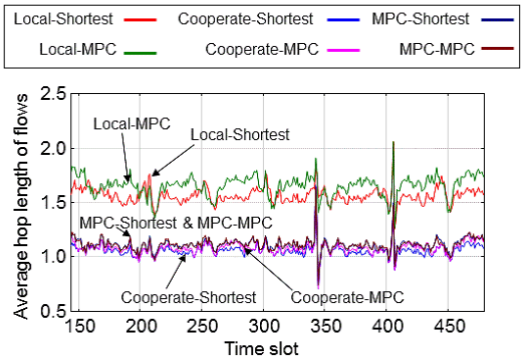
In content delivery services, cache-server selection and delivery-route control are independently operated by CDN

TABLE III
AVERAGE NUMBER OF FAILED REQUESTS IN VERIO NETWORK

Route selection	Cache-server selection		
	Local	Cooperate	MPC
Shortest	1980.5	316.9	12.41
MPC	985.4	316.9	11.7



(a) CAIS network



(b) Verio network

Fig. 6. Average hop length of delivery flows in each time slot

TABLE IV
AVERAGE HOP LENGTH OF DELIVERY FLOWS IN CAIS NETWORK

Route selection	Cache-server selection		
	Local	Cooperate	MPC
Shortest	3.93	1.99	2.14
MPC	4.10	2.01	2.14

TABLE V
AVERAGE HOP LENGTH OF DELIVERY FLOWS IN VERIO NETWORK

Route selection	Cache-server selection		
	Local	Cooperate	MPC
Shortest	1.56	1.06	1.11
MPC	1.65	1.08	1.11

provider and ISP, respectively. However, the number of ISPs providing the CDN service has been increasing, and these ISPs providing CDN can control the cache-sever and the delivery-route selection simultaneously. We proposed a method of optimizing server selection using the MPC, which has been widely used in system control. We also proposed a method of simultaneously optimizing both the cache-server and delivery-route selection using the MPC. Through numerical evaluation using the two actual network topologies of Tier-1 ISPs and the access log data of a VoD service, we confirmed that we could achieve almost the same effect by optimizing only cache-server selection using the MPC, compared with optimizing both operations simultaneously. In hub & spoke type networks, in which hub nodes existed, the reduction effect of the failed-request count obtained by using the MPC as the cache-server selection was especially high because many cache servers were probabilistically selected, and the diversity of selected cache servers was high when using the MPC. The number of failed requests that were not accommodated in the network can be reduced by about 1/30 with the proposed two methods using the MPC.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Internal Affairs and Communications of Japan.

REFERENCES

[1] B. Ahlgren, et al., A Survey of Information-Centric Networking," IEEE Commun. Mag., vol.50, no.7, pp.26-36, July 2012.
[2] M. Ahmed, S. Spagna, F. Huici, and S. Niccolini, A Peek into the Future: Predicting the Evolution of Popularity in User Generated Content, ACM WSDM 2013.
[3] D. Applegate, A. Archer, V. Gopalakrishnan, Optimal Content Placement for a Large-Scale VoD System, ACM CoNEXT 2010.
[4] T. Benson, A. Anand, A. Akella, and M. Zhang, MicroTE: Fine Grained Traffic Engineering for Data Centers, ACM CoNEXT 2011.
[5] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho, Seven Years and One Day: Sketching the Evolution of Internet Traffic, IEEE INFOCOM 2009.
[6] Y. Borghol, et al., Characterizing and Modeling Popularity of User-generated Videos, ACM Performance Evaluation, Vol.68, No.11, pp. 1037-1055, 2011.
[7] CAIDA, http://www.caida.org/tools/visualization/mapnet/Data/
[8] G. Chatzopoulou, C. Sheng, M. Faloutsos, A first step towards understanding popularity in YouTube, IEEE Global Internet Symposium 2010.
[9] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi, A Survey on Content-Oriented Networking for Efficient Content Delivery, IEEE Commun. Mag., vol.49, no.3, pp.121-127, Mar. 2011.
[10] Cisco Visual Networking Index: Forecast and Methodology, 2013-2018.
[11] E. Danna, S. Mandal, A. Singh, A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering, IEEE INFOCOM 2012.
[12] D. DiPalantino and R. Johari, Traffic Engineering vs. Content Distribution: A Game Theoretic Perspective, IEEE INFOCOM 2009.
[13] A. Elwalid, C. Jin, S. Low, and I. Widjaja, MATE: MPLS Adaptive Traffic Engineering, IEEE INFOCOM 2001.
[14] B. Frank, et al., Pushing CDN-ISP Collaboration to the Limit, ACM SIGCOMM Computer Communication Review, Vol.43, NO.3, pp.35-44, 2013.
[15] B. Fortz and M. Thorup, Internet Traffic Engineering by Optimizing OSPF Weights, IEEE INFOCOM 2000.
[16] B. Fortz and M. Thorup, Robust optimization of OSPF/IS-IS weights, INOC 2003.
[17] G. Gursun, M. Crovella, and I. Matta, Describing and Forecasting Video Access Patterns, IEEE INFOCOM 2011 Mini.
[18] T. He, D. Goeckel, R. Raghavendra, and D. Towsley, Endhost-Based Shortest Path Routing in Dynamic Networks: An Online Learning Approach, IEEE INFOCOM 2013.
[19] C. Y. Hong, et al., Achieving high utilization with software-driven WAN, ACM SIGCOMM 2013.
[20] V. Jacobson, et al., Networking Named Content, ACMCoNEXT 2009.
[21] S. Jain, et al., B4: Experience with a globally deployed software defined WAN, ACM SIGCOMM 2013.
[22] W. Jiang, R. Shen, J. Rexford, and M. Chiang, Cooperative Content Distribution and Traffic Engineering in an ISP Network, ACM SIGMETRICS 2009.
[23] N. Kamiyama, Y. Takahashi, K. Ishibashi, K. Shiomoto, T. Otoshi, Y. Ohsita, and M. Murata, Flow Aggregation for Traffic Engineering, IEEE GLOBECOM 2014.
[24] S. Kandula, D. Katabi, B. Davie, and A. Charny, Walking the Tightrope: Responsive Yet Stable Traffic Engineering, ACM SIGCOMM 2005.
[25] C. Labovitz, S. Iekel-Johnson, J. Oberheide, and F. Jahanian, Internet Inter-Domain Traffic, ACM SIGCOMM 2010.
[26] E. Nygren, R. Sitaraman, and J. Sun, The Akamai Network: A Platform for High-Performance Internet Applications, ACM SIGOPS 2010.
[27] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, N. Kamiyama, K. Ishibashi, K Shiomoto, and T. Hashimoto, Traffic Engineering Based on Model Predictive Control, IEICE Transactions on Communications (to appear).
[28] J. Ott, M. Sanchez, J. Rula, F. Bustamante, Content Delivery and the Natural Evolution of DNS, ACM IMC 2012.
[29] I. Poese, B. Frank, B. Ager, G. Smaragdakis, and A. Feldmann, Improving Content Delivery Using Provider-aided Distance Information, ACM IMC 2010.
[30] I. Poese, B. Frank, G. Smaragdakis, S. Uhlig, A. Feldmann, and B. Maggs, Enabling Content-aware Traffic Engineering, ACM SIGCOMM Computer Communication Review, Vol. 42, NO. 5, pp. 22-28, 2012.
[31] S. J. Qin and T. A. Badgwell, A survey of industrial model predictive control technology, Control Engineering Practice, vol. 11, pp. 733—764, July 2003.
[32] G. Rossini and D. Rossi, Coupling Caching and Forwarding: Benefits, Analysis, and Implementation, ACM ICN 2014.
[33] A. Sharma, A. Mishra, V. Kumar, A. Venkataramani, Beyond MLU: An Application-Centric Comparison of Traffic Engineering Schemes, IEEE INFOCOM 2011.
[34] A. Su, D. Choffnes, A. Kuzmanovic, and F. Bustamante, Drafting Behind Akamai: Inferring Network Conditions Based on CDN Redirections, ACM Trans. Networking, 17(6), pp. 1752-1765, 2009.
[35] X. Tang and J. Xu, On Replica Placement for QoS-Aware Content Distribution, IEEE INFOCOM 2004.
[36] H. Yu, D. Zheng, B. Zhao, and W. Zheng, Understanding User Behavior in Large-Scale Video-on-Demand Systems, ACM EuroSys 2006.