PAPER Special Section on the Architectures, Protocols, and Applications for the Future Internet

# Adaptive TTL Control to Minimize Resource Cost in Hierarchical Caching Networks

**Satoshi IMAI**[†,††a)], **Kenji LEIBNITZ**[†††b)], ***Members***, ***and*** **Masayuki MURATA**[††c)], ***Fellow***

**SUMMARY** Content caching networks like *Information-Centric Networking* (ICN) are beneficial to reduce the network traffic by storing content data on routers near to users. In ICN, it becomes an important issue to manage system resources, such as storage and network bandwidth, which are influenced by cache characteristics of each cache node. Meanwhile, cache aging techniques based on *Time-To-Live* (TTL) of content facilitate analyzing cache characteristics and can realize appropriate resource management by setting efficient TTLs. However, it is difficult to search for the efficient TTLs in a distributed cache system connected by multiple cache nodes. Therefore, we propose an adaptive control mechanism of the TTL value of content in distributed cache systems by using predictive models which can estimate the impact of the TTL values on network resources and cache performance. Furthermore, we show the effectiveness of the proposed mechanism.

***key words:*** *cache aging, Time-To-Live, content dissemination, in-network caching, content-centric networking, Newton's method*

## 1. Introduction

The recently increasing network traffic, due to the spread of various network applications, boosts demand for efficiently distributing contents of each application/service. Meanwhile, *Content Delivery Networks* (CDNs) can provide content delivery at the edge of the networks by allocating content replicas in cache servers, which are in geographical proximity to users, and it is expected that CDNs can efficiently reduce the network traffic. Furthermore, various caching architectures [1] for *Information-Centric Networking* (ICN), in which each node has caching functionality for content, have been actively discussed. In *Content-Centric Networking* (CCN) [2] for ICN, content publishers advertise newly released content from the origin site along predefined routes. A content request is forwarded to each content router until the requested content is found. When the requested content is found on a content router, the content data are transmitted on the reverse route along the request forwarding route. Moreover, content data are cached on all cache nodes along the transmission route based on the specific caching policy. Therefore, distributed mechanisms like CCN can automate the placement and delivery of content data for efficient content delivery.

Recently, network virtualization technologies such as *Network Function Virtualization* (NFV) have been attracting attention [3], [4]. On the assumption that ICN services are provided on the network virtualization platform, the caching services can manage the resource cost like monetary cost for each content/application by providing necessary system resources, such as storage and communication bandwidth, in the caching network.

In the distributed cache mechanisms, the network traffic and cache characteristics are influenced by the cache locations and depend on the memory size in each cache node on the delivery tree. Therefore, it is difficult to manage the relationship among resource cost, network resources, such as memory and network devices, and cache characteristics, such as cache hit ratio.

Meanwhile, the persistent storage of content in each cache node is inefficient because the content, such as video streaming, generally has a limited lifetime. Therefore, caching mechanisms often use a limited period of time, called *Time-To-Live* (TTL) for storing content. Moreover, TTL-based caching facilitates analyzing the cache characteristics and provides flexibility and adaptability of cache management per content [5]–[7].

In this paper, we first introduce predictive models which estimate the impact of TTL on cache characteristics, network resources, and the resource cost in the distributed cache system. Furthermore, we propose an adaptive TTL control mechanism based on the predictive models to reduce the total resource cost and demonstrate the effectiveness of the proposed mechanism by evaluations.

Our contributions in this paper are as follows:

- We propose an adaptive control mechanism using TTL to reduce the total resource cost in a distributed cache system.
- We highlight the control methodology for general hierarchical TTL-based caching as an application of Newton's method [8] for finding the optimum of a nonlinear function.

The remainder of this paper is organized as follows. Section 2 discusses issues in distributed caching and an out-

Manuscript received May 30, 2014.
Manuscript revised October 3, 2014.
Manuscript publicized December 11, 2014.

[†]The author is with Fujitsu Laboratories Ltd., Kawasaki-shi, 211–8588 Japan.
[††]The authors are with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565–0871 Japan.
[†††]The author is with Center for Information and Neural Networks (CiNet), National Institute of Information and Communications Technology, and Osaka University, Suita-shi, 565–0871 Japan.
a) E-mail: imai.satoshi@jp.fujitsu.com
b) E-mail: leibnitz@nict.go.jp
c) E-mail: murata@ist.osaka-u.ac.jp
DOI: 10.1587/transinf.2014NTP0011

line of approach proposed in this paper followed by Sect. 3 which summarizes related work. We propose a system model in Sect. 4 and an adaptive TTL controller in Sect. 5. Furthermore, we present evaluation results using the proposed model in Sect. 6. Finally, we conclude the paper in Sect. 7.

## 2. Issues and Approach

Most caching networks have hierarchical tree structures, logically or physically connected by cache nodes (*CN*s) [5], [9], [10] and each content is delivered on the caching hierarchy rooted at each origin site of content. Likewise in CCN, each content router autonomously constructs a caching hierarchy rooted at an origin site. In these distributed mechanisms, the caching hierarchy is constructed by routes between the origin site, caching *CN*s, and users, such that less popular content is cached on *CN*s near to the origin site and more popular content is cached on *CN*s near to users.

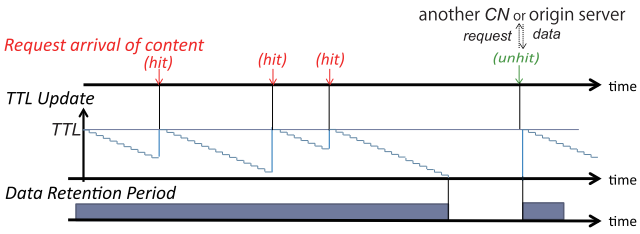
In order to efficiently use system resources in the distributed cache systems, the appropriate cache allocation should be executed. Additionally, on the assumption that network devices and memory can be used in proportion to their usage based on the principle of network virtualization, the cache system should consider the *storage cost* for storing the content and the *bandwidth cost* for using network devices when data are transmitted.

This paper assumes that each *CN* judges whether to cache data by using a counter to manage TTL of content. Each *CN* discards the content with expired TTL and forwards new requests for the discarded content to another *CN* or the origin server. After receiving the delivered content, it caches the data of the content again and updates the counter based on one of the following policies in TTL-based caching [5]–[7].

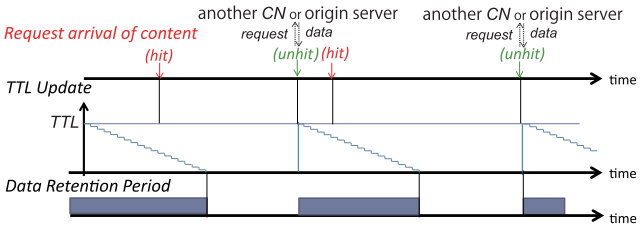
- *Policy 1:* Each *CN* resets the time counter to the TTL value for each content every time a new request for this content arrives and decreases the counter by 1 every time unit (cf. Fig. 1 (a)).
- *Policy 2:* Each *CN* resets the time counter to the TTL value for each content only when there are unsuccessful requests and decreases the counter by 1 every time unit (cf. Fig. 1 (b)).

As shown in Fig. 2, in general distributed TTL-based cache systems, *storage cost* increases and *bandwidth cost* decreases as the TTL value of content increases. In order to reduce the total resource cost, we should consider that there is a tradeoff for the TTL value between *storage cost* and *bandwidth cost*. Therefore in this paper, we propose an adaptive control mechanism to search for the TTL of each content minimizing the total resource cost in the hierarchical cache system.

While in real systems usually content objects are split into multiple chunks, our proposed cache mechanism considers only entire content objects. However, the methodology can be extended to chunk level caching on the analogy of object level caching. Meanwhile to manage the TTL value per chunk, the system must handle much information about the TTL values. Furthermore, it is necessary to establish a new architecture for managing chunks with TTL values, which is under discussion in the research field [7], [11]–[13] of ICN. For instance, in the architecture for managing the TTL value per chunk, the chunks may be categorized into some classes with almost the same TTL value, or chunks of an object may have the same TTL value. The implementation of extending our mechanism to chunk level caching depends on this architecture. In this paper. we consider the extension of the proposed mechanism to chunk level caching as out of scope and we will further study it as another work.




(a) *Policy 1*




(b) *Policy 2*

**Fig. 1** TTL-based caching.

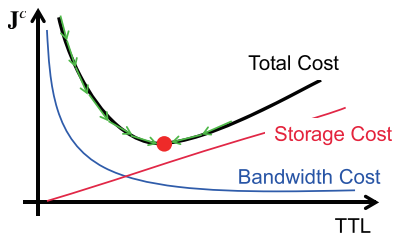



**Fig. 2** The tradeoff between *storage cost* and *bandwidth cost*.

## 3. Related Work

The modeling of efficient memory management is a major issue in content caching systems. Traditionally, there are content placement algorithms [14], [15] which minimize the cost imposed for content storage and queries, or maximize performance such as proximity to content. In contrast to the above-mentioned content placement problems, Borst *et al.* [10] formulate a linear programming model based on

a hierarchical structure for content locations to minimize bandwidth costs through a distributed solution.

In view of energy efficiency for content delivery networks, Guan *et al.* [16] propose a design guideline for cache allocation in consideration of the tradeoff between traffic transmission power and caching power for content delivery architectures such as "Conventional and decentralized server-based CDN", "Centralized server-based CDN using dynamic optical bypass", and CCN. Furthermore, we proposed in our previous work [17] a new *Integer Linear Programming* (ILP) model to design the most energy efficient cache locations in consideration of the multiplexed caching hierarchies and a distributed cache mechanism to locally search for energy efficient cache locations attempting to be close to the optimal cache locations.

Carofiglio *et al.* [11] explore the impact of storage management on the cache performance per application in CCN and evaluate the effectiveness of static storage partitioning and dynamic management by priority-based weighted fair queuing schemes combined with TTL-based caching. Hou *et al.* [12] propose an analytical model of a hierarchical TTL-based caching system. The proposed mechanism updates content objects by using the TTL values which are randomly set for each node according to its layer on the delivery tree. Fofack *et al.* [13] propose a statistical model for some network topologies connected by cache nodes using *Policy 1* in TTL-based caching under the assumptions that requests are generated by renewal processes. Additionally, Berger *et al.* [7] also model cache characteristics of *Policy 1*, *Policy 2*, and the combination of both in TTL-based caching. Furthermore in [18], we proposed an analytical model using simple matrix equations, which can analyze cache performance and request propagation from all requesting users.

However, these proposals don't discuss adaptive resource control to reduce the resource cost in hierarchical caching by searching for an appropriate TTL of content. Therefore, we propose a TTL control mechanism based on the proposed predictive models to reduce the total resource cost consisting of *storage cost* and *bandwidth cost*.

## 4. System Model

In order to model the cache characteristics in a hierarchical cache system, we first introduce the following model [5]–[7] of the cache probability of content $c$ having the request rate $\lambda^c$ to a *CN*.

- *Policy 1*: $$\boldsymbol{f}(\lambda^c, TTL^c) = 1 - e^{-\lambda^c TTL^c} \quad (1)$$
- *Policy 2*: $$\boldsymbol{f}(\lambda^c, TTL^c) = \frac{\lambda^c TTL^c}{1 + \lambda^c TTL^c} \quad (2)$$

The parameter $TTL^c$ is the TTL value of content $c$. These statistical functions can provide a good approximation of the cache hit ratio of content at a *CN* under the assumption that the inter-arrival time of content requests follows an exponential distribution.

### 4.1 Request Propagation Model

In [18], we proposed an analytical model using *Policy 1* of the hierarchical cache system in which each *CN* caches content data delivered by another *CN* or an origin server when the TTL counter is above 0 and discards the content when the counter becomes 0.

The proposed model is expressed by simple matrix equations combining the statistical model of each *CN* in Eqs. (1) or (2) and can predict the cache characteristics and resource usage in hierarchical caching when the TTL value of content is given.

Here, we show the evaluation model to analyze the cache characteristics of the hierarchical cache system based on *Policy 1* and *Policy 2* in TTL-based caching. The propagation of each request of content $c$ on its caching hierarchy is expressed by the following model.

$$\boldsymbol{\Lambda}^c[s+1] = \mathrm{D}^c[s] \cdot \boldsymbol{\Lambda}^c[s] + \mathbf{R}^c, \forall c \quad (3)$$

Under the condition that $M$, $N$, and $s$ are the number of *CN*s, the number of sites having requesting users, and the number of steps that each request propagates to the next *CN*, respectively, we define $\boldsymbol{\Lambda}^c$ as the $M \times N$ matrix consisting of the request rates $\lambda^c_{(i,j)}$ of content $c$ forwarded from the requesting user in site $j$ to $CN_i$ and $\mathbf{R^c}$ as the $M \times N$ matrix of which elements are the request rates $r^c_i$ of content $c$ input directly from users in site $j$ to $CN_i$.

We define $\mathrm{D}^c$ as the $M \times M$ matrix of request propagations for content $c$. The $(m, n)$ element of the matrix $\mathrm{D}^c$ if $CN_m$ is a parent node of $CN_n$ is defined as:

$$[\mathrm{D^c}]_{(m,n)} := 1 - \boldsymbol{f}\Big(\sum_j^N \lambda^c_{(n,j)}, TTL^c\Big) \quad (4)$$

otherwise, $[\mathrm{D^c}]_{(m,n)} := 0$. In Fig. 3, we show an example of these matrices for the delivery tree having 5 nodes and origin 1.

With Eq. (3), we can predict the request propagation process for content requested from each site. Moreover, the steady state of propagated request rates per content can be derived by iteratively calculating the equation $s_{max}$-times, which is the maximum number of hops from each site having requesting users to its origin site.

### 4.2 Resource Cost Model

We next define the resource cost $\mathrm{J}^c(TTL^c, \lambda^c_{(m,n)})$ of content $c$ in the hierarchical cache system when the TTL value is set to $TTL^c$ as follows.

$$\mathrm{J}^c(TTL^c, \lambda^c_{(m,n)}) = \theta_c C_s [1, \cdots, 1]_{1\times M} \begin{bmatrix} \boldsymbol{f}(\sum_j^N \lambda^c_{(1,j)}, TTL^c) \\ \vdots \\ \boldsymbol{f}(\sum_j^N \lambda^c_{(M,j)}, TTL^c) \end{bmatrix}$$

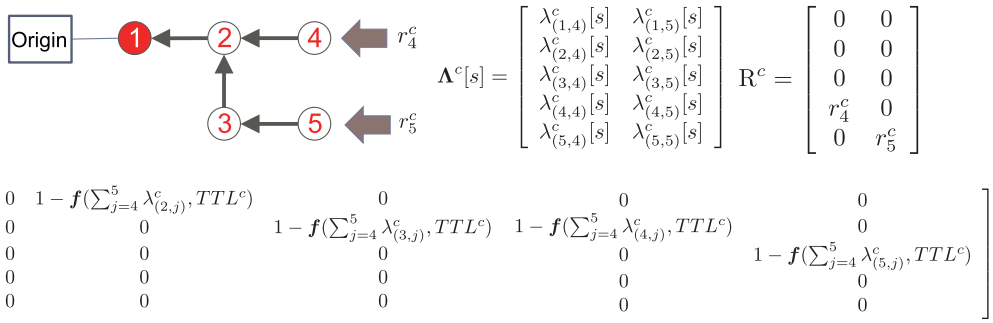



**Fig. 3** An example of matrices $\mathbf{\Lambda}^c$, $\mathbf{R}^c$, and $\mathrm{D}^c$ for the request propagation on the delivery tree having origin 1.

$$+\theta_c C_b[1, \cdots, 1]_{1\times N} \begin{bmatrix} Tr_1^c \\ \vdots \\ Tr_N^c \end{bmatrix}. \tag{5}$$

The first and the second terms are the total cost for storing and transmitting data of content $c$ in the network, which use $\theta_c$, $C_s$, and $C_b$ as data size of content $c$ [bit], the *storage cost* [cost/bit] and *bandwidth cost* [cost/bit], respectively.

The cumulative number of traffic flows $Tr_j^c$ through each *CN* on the delivery route for content $c$ having origin site $o$ requested by users in site $j$ is shown as follows.

$$\begin{aligned} \mathbf{Tr}^c &:= [Tr_1^c \cdots Tr_j^c \cdots Tr_N^c]^{\mathrm{T}} \\ &= (\mathbf{Hp} * \mathbf{\Lambda}^c)^{\mathrm{T}} \begin{bmatrix} \boldsymbol{f}(\sum_j^N \lambda^c_{(1,j)}, TTL^c) \\ \vdots \\ \boldsymbol{f}(\sum_j^N \lambda^c_{(M,j)}, TTL^c) \end{bmatrix} \\ &\quad + (\mathbf{Hp}[o,] * \mathbf{\Lambda}^c[o,])^{\mathrm{T}} (1 - \boldsymbol{f}(\sum_j^N \lambda^c_{(o,j)}, TTL^c)) \end{aligned} \tag{6}$$

Here, we define "$*$" as the element-wise product (Hadamard product) of a matrix or vector and $\mathbf{Hp} = [h_{(i,j)}]_{M\times N}$ as the matrix consisting of shortest hop length $h_{(i,j)}$ from $CN_i$ to $CN_j$ and the terms $\mathbf{Hp}[o,]$ and $\mathbf{\Lambda}^c[o,]$ are the $o$-th row vectors in matrices of $\mathbf{Hp}$ and $\mathbf{\Lambda}^c$, respectively. Moreover, the second term in Eq. (6) presents the amount of transmitted data which aren't cached on the network.

## 5. Proposed TTL Controller

In this paper, we propose a TTL controller which adaptively updates TTL of content to reduce the total resource cost in the hierarchical cache system.

On the assumption that the control interval for TTL is set to $\Delta$ sec (e.g. 10 sec), the proposed controller updates TTL of content $c$ at each node by a manipulated value $\delta t_c$ (e.g. 1 sec), in time unit $k$ to reduce the total resource cost in Eq. (5) as Eq. (7).

$$TTL_{k+1}^c = \Theta\left(TTL_k^c + \phi(\delta t_c)\right) \tag{7}$$

The terms $\phi(\cdot)$ and $\Theta(\cdot)$ are a manipulation function and a saturation function using the maximum TTL value as follows.

$$\phi(\delta t_c) = -\mathrm{sgn}(d\mathrm{J}^c/dTTL)\delta t_c$$
$$\Theta(x) = \max(0, \min(\mathrm{maxTTL}, x))$$

The term $\mathrm{sgn}(\cdot)$ is the sign function. Here, the cost function of $\mathrm{J}^c$ is downwardly convex over the TTL value and the proposed controller searches for the TTL value of content minimizing the total cost by decreasing slope of the differential $d\mathrm{J}^c/dTTL^c$.

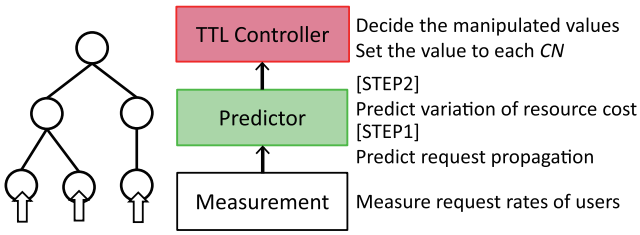



**Fig. 4** Outline of the proposed control mechanism.

### 5.1 Design Algorithm of TTL Controller

In the proposed control mechanism as shown in Fig. 4, the TTL controller of content is designed by the following two steps.

- **STEP1:** Predict the equilibrium of propagated request rates at each *CN* in Eq. (3) and the variation of the propagated requests when the TTL value of content is increased by $\delta t_c$.
- **STEP2:** Predict the variation of the resource cost when the TTL value of content is increased by $\delta t_c$, which is the differential of the cost function derived from the predictive results in **STEP1**.

All variables in the proposed model are summarized in Table 1. Next, we show the design process of the proposed TTL controller.

#### 5.1.1 Prediction of Equilibrium and Variation of Propagated Request Rates

The equilibrium of propagated requests $\mathbf{\Lambda}^c$ of content $c$ to

**Table 1** Variables used in the proposed model.

| Variable | Type | Definition |
|---|---|---|
| $M$ | given | number of *CN*s |
| $N$ | given | number of sites having requesting users |
| $\theta_c$ | given | data size of content $c$ [bit] |
| $C_s$ | given | storage cost [cost/bit] |
| $C_b$ | given | data transmission cost [cost/bit] |
| **Hp** | given | matrix consisting of hop length $h_{(i,j)}$ from $CN_i$ to $CN_j$ |
| $r_i^c$ | measured | request rate of content $c$ requested by users in site $i$ |
| $TTL^c$ | control | TTL of content $c$ |
| $\delta t_c$ | given | manipulated value of TTL control |
| maxTTL | given | maximum value of TTL |
| $\lambda_{(i,j)}^c$ | predicted | propagated request rates to $CN_i$ for content $c$ requested by users in site $j$ |
| $\delta\lambda_{(i,j)}^c$ | predicted | variation of propagated request rates to $CN_i$ for content $c$ requested by users in site $j$ |
| $\mathrm{Tr}_j^c$ | predicted | cumulative number of traffic flows through each *CN* on the delivery route for content $c$ requested by users in site $j$ |
| $\omega_{(i,j)}^c$ | predicted | request rates for content $c$ requested by users in site $j$ forwarding from $CN_i$ to its parent *CN* |
| $\delta\omega_{(i,j)}^c$ | predicted | variation of request rates for content $c$ requested by users in site $j$ forwarding from $CN_i$ to its parent *CN* |

each *CN* on its caching hierarchy satisfies the following matrix equation:

$$\mathbf{\Lambda}^c = \mathrm{D}^c\mathbf{\Lambda}^c + \mathbf{R}^c, \forall c \tag{8}$$

where $\mathrm{D}^c$ is the equilibrium of the request propagation matrix for content $c$ in $TTL^c$. These can be derived by iterative calculation of Eq. (3).

Furthermore, we predict the variation of the propagated request rates $\delta\mathbf{\Lambda}^c$ when $TTL_k^c$ is increased by $\delta t_c$ as follows.

$$\begin{aligned}&\mathbf{\Lambda}^c + \delta\mathbf{\Lambda}^c = (\mathrm{D}^c + \delta\mathrm{D}^c)(\mathbf{\Lambda}^c + \delta\mathbf{\Lambda}^c) + \mathbf{R}^c, \forall c\\ &\quad\Leftrightarrow\ \delta\mathbf{\Lambda}^c \approx \mathrm{D}^c\delta\mathbf{\Lambda}^c + \delta\mathrm{D}^c\mathbf{\Lambda}^c, \forall c\end{aligned}$$

The $(m,n)$ element of the matrix $\delta\mathrm{D}^c$ if $CN_m$ is a parent node of $CN_n$ is defined as follows.

$$\begin{aligned}[\delta\mathrm{D}^c]_{(m,n)} := &-\frac{\partial \boldsymbol{f}(\sum_j^N \lambda_{(n,j)}^c, TTL^c)}{\partial TTL}\delta t\\ &-\sum_h^N \frac{\partial \boldsymbol{f}(\sum_j^N \lambda_{(n,j)}^c, TTL^c)}{\partial \lambda_{(n,h)}^c}\delta\lambda_{(n,h)}^c\end{aligned}$$

Otherwise, $[\delta\mathrm{D}^c]_{(m,n)} := 0$. The matrix $\delta\mathbf{\Lambda}^c$ is defined as

$$\delta\mathbf{\Lambda}^c := [\delta\lambda_{(i,j)}^c]_{M\times N}.$$

As a result, we can derive the following equation with the matrices $\mathrm{G}^c$ and $\mathrm{H}^c$.

$$\delta\mathbf{\Lambda}^c \approx \mathrm{G}^c\delta\mathbf{\Lambda}^c + \mathrm{H}^c\mathbf{\Lambda}^c\delta t_c. \tag{9}$$

For the two policies in TTL-based caching, the $(m,n)$ elements of the matrices $[\mathrm{G}^c]_{(m,n)}$ and $[\mathrm{H}^c]_{(m,n)}$ if $CN_m$ is a parent node of $CN_n$ can be derived as follows.

- *Policy 1*:

$$[\mathrm{G}^c]_{(m,n)} := \left(1 - TTL^c\sum_j^N \lambda_{(n,j)}^c\right)e^{-TTL^c\sum_j^N\lambda_{(n,j)}^c}$$

$$[\mathrm{H}^c]_{(m,n)} := -\sum_j^N \lambda_{(n,j)}^c e^{-TTL^c\sum_j^N\lambda_{(n,j)}^c}$$

- *Policy 2*:

$$[\mathrm{G}^c]_{(m,n)} := \frac{1}{1+TTL^c\sum_j^N\lambda_{(n,j)}^c} - \frac{TTL^c\sum_j^N\lambda_{(n,j)}^c}{(1+TTL^c\sum_j^N\lambda_{(n,j)}^c)^2}$$

$$[\mathrm{H}^c]_{(m,n)} := -\frac{\sum_j^N\lambda_{(n,j)}^c}{(1+TTL^c\sum_j^N\lambda_{(n,j)}^c)^2}$$

In both cases, $[\mathrm{G}^c]_{(m,n)} = 0$ and $[\mathrm{H}^c]_{(m,n)} = 0$ if $CN_m$ is not a parent node of $CN_n$.

Because the square matrix $(I - \mathrm{G}^c)$ always has full rank, we can derive the following equation.

$$\delta\mathbf{\Lambda}^c = (I - \mathrm{G}^c)^{-1}\mathrm{H}^c\mathbf{\Lambda}^c\delta t_c \tag{10}$$

#### 5.1.2 Prediction of Variation of Resource Cost

For the total resource cost $\mathrm{J}^c$ in Eq. (5), we can differentiate the cost function $\delta\mathrm{J}^c(TTL^c, \delta t_c, \mathbf{\Lambda}, \delta\mathbf{\Lambda})$ as shown in the Appendix, which is the variation of the total resource cost $\mathrm{J}^c$ when $TTL^c$ is increased by $\delta t_c$. By substituting $TTL_k^c$ and $\mathbf{\Lambda}^c$ in Eq. (8), and $\delta\mathbf{\Lambda}^c$ in Eq. (10) for the differential function $\delta\mathrm{J}^c$, we can derive the following linear equation of $\delta t_c$ where $Q^c$ is a scalar.

$$\delta\mathrm{J}^c = Q^c\delta t_c$$

Therefore, the TTL controller in Eq. (7) can be derived by

$$\begin{aligned}&TTL_{k+1}^c = \Theta\left(TTL_k^c + \phi(\delta t_c)\right),\\ &\phi(\delta t_c) = -\mathrm{sgn}(Q^c)\delta t_c,\end{aligned} \tag{11}$$

which means that the TTL value of content is updated at each *CN* by the following rule. When $Q^c$ is negative/positive, $TTL^c$ is increased/decreased by $\delta t_c$ and when $Q^c$ is zero, $TTL^c$ is not updated.

The calculation order per content in the proposed prediction process is $O(M^2N)$, but we can divide the predictive process into local prediction processes of each *CN* with calculation order of $O(N)$.

### 5.2 Distributed Prediction

The proposed algorithm of the TTL controller can be divided into recursive processes at each *CN* which coordinates *CN*s on the delivery tree every time unit $\Delta$ (cf., Fig. 5).

In the proposed mechanism, the tracking performance

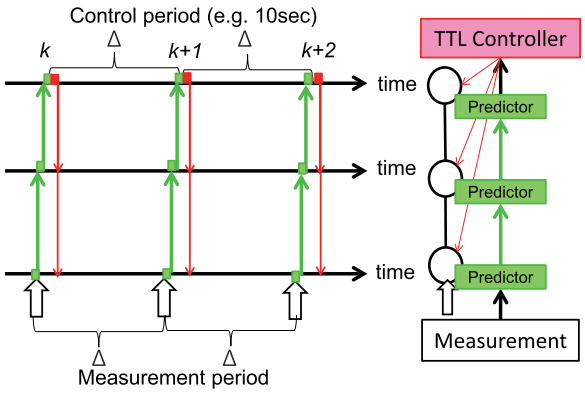



**Fig. 5** Distributed prediction-based control mechanism.

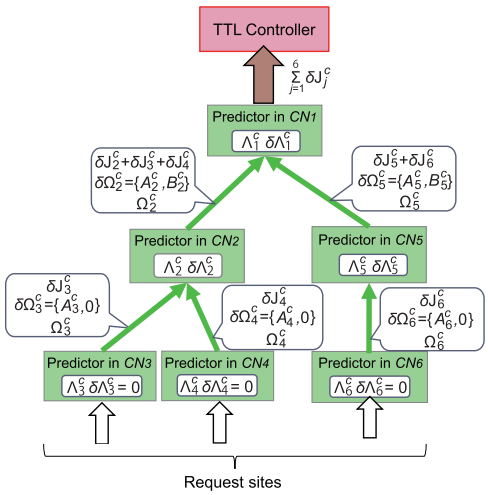



**Fig. 6** An example of the distributed prediction.

to change the request rates becomes better as the control interval $\Delta$ is smaller. However, the control interval should be set as a long period to measure the statistics of request rates. Meanwhile, Santos *et al.* [19] proposed a statistics table for CCN, which can online estimate request rates by using the exponentially weighted moving average (EWMA), to implement their proposed Content-Centric Dissemination Algorithm (CEDO) for maximizing the total delivery throughput. Therefore, we can also use EWMA to online measure the request rates. In this paper, we assume that the exchange of information between *CN*s and TTL controllers, including measured request rates and manipulated TTL values for each content, is bundled in a single measurement/control message.

In the distributed mechanism as shown in Fig. 6, $CN_m$ predicts the propagated request rate $\mathbf{\Lambda}_m^c = \lambda_{(m,j)}^c, \forall j, c$ in Eq. (8), the variation of request rate $\boldsymbol{\delta\Lambda}_m^c = \delta\lambda_{(m,j)}^c, \forall j, c$ in Eq. (10), and the variation of the total resource cost $\delta \mathrm{J}_m^c$ in Eq. (A· 2) of the Appendix in the order from *CN*s (leaves) on the bottom layer to a *CN* (root) on the upper layer in the delivery tree.

### 5.2.1 Prediction of Equilibrium and Variation of Propagated Request Rates

$CN_m$ first calculates the information $\Omega_m^c$, which is the predictive amount of propagated request rates from $CN_m$ to its parent $CN_p$, by the product of the matrix element $\mathrm{D}^c[p,m]$ and the $m$-th row vector $\Lambda_m^c$ in the right side of Eq. (8).

$$\Omega_m^c = \left(1 - \boldsymbol{f}(\sum_j^N \lambda_{(m,j)}^c, TTL^c)\right)\Lambda_m^c$$

Furthermore, $CN_m$ can predict the equilibrium of propagated request rates $\Lambda_m^c$ by using its children information on the delivery tree,

$$\Lambda_m^c = \sum_{h \in \mathrm{Child}_m} \Omega_h^c + \mathrm{R}_m^c \tag{12}$$

where $\mathrm{R}_m^c$ is the request rate input directly to $CN_m$ by users and corresponds to the $m$-th row vector of matrix $\mathbf{R}^c$ in Eq. (8). The terms $\Omega_m^c$ and $\Lambda_m^c$ are both $1 \times N$ vectors and $\mathrm{Child}_m$ is a set of *CN*s directly under $CN_m$.

Furthermore, $CN_m$ calculates $\delta\Omega_m^c$, which is the variation of the request rate from $CN_m$ to its parent $CN_p$ when increasing $TTL$ by $\delta t_c$. It is calculated by the sum of the product of the matrix element $\mathrm{G}^c[p,m]$ and vector $\delta\Lambda_m^c$ and the product of the matrix element $\mathrm{H}^c[p,m]$ and vector $\Lambda_m^c$ in the right side of Eq. (9). Therefore, the information becomes a linear function of $\delta t_c$ with $A_m^c$ and $B_m^c$ having $\Lambda_m^c$ in Eq. (12) and $\delta\Lambda_m^c$ derived by the children information $\delta\Omega_h^c$ of $CN_m$

$$\delta\Omega_m^c = A_m^c \delta t_c + B_m^c$$

where $A_m^c$ and $B_m^c$ are $1 \times N$ vectors which are defined for both policies as follows.

- *Policy 1*:

$$A_m^c = -\left(\sum_j^N \lambda_{(m,j)}^c e^{-TTL^c \sum_j^N \lambda_{(m,j)}^c}\right)\Lambda_m^c$$

$$B_m^c = \left(\left(1 - TTL^c \sum_j^N \lambda_{(m,j)}^c\right)e^{-TTL^c \sum_j^N \lambda_{(m,j)}^c}\right)\delta\Lambda_m^c$$

- *Policy 2*:

$$A_m^c = -\left(\frac{\sum_j^N \lambda_{(m,j)}^c}{(1 + TTL^c \sum_j^N \lambda_{(m,j)}^c)^2}\right)\Lambda_m^c$$

$$B_m^c = \left(\frac{1}{(1 + TTL^c \sum_j^N \lambda_{(m,j)}^c)^2}\right)\delta\Lambda_m^c$$

Moreover, $CN_m$ predicts the variation of propagated request rates $\Lambda_m^c$ by using its children information $\delta\Omega_h^c$,

$$\delta\Lambda_m^c = \sum_{h \in \mathrm{Child}_m} \delta\Omega_h^c \tag{13}$$

- *Policy 1*:

$$\delta \mathrm{J}_m^c(TTL^c, \delta t_c, \mathbf{\Lambda}_m^c, \delta \mathbf{\Lambda}_m^c) = \theta_c C_s \left( \sum_j^N \lambda_{(m,j)}^c e^{-TTL^c \sum_j^N \lambda_{(m,j)}^c} \delta t_c + TTL^c e^{-TTL^c \sum_j^N \lambda_{(m,j)}^c} \sum_h^N \delta \lambda_{(m,h)}^c \right)$$
$$+\theta_c C_b \left( \sum_j^N h_{(m,j)} \delta \lambda_{(m,j)}^c (1 - e^{-TTL^c \sum_j^N \lambda_{(m,j)}^c}) + \sum_j^N h_{(m,j)} \lambda_{(m,j)}^c \left( \sum_j^N \lambda_{(m,j)}^c e^{-TTL^c \sum_j^N \lambda_{(m,j)}^c} \delta t_c + TTL^c e^{-TTL^c \sum_j^N \lambda_{(m,j)}^c} \sum_h^N \delta \lambda_{(m,h)}^c \right) \right)$$
$$+ \begin{cases} 0 & \text{if } m \neq o \\ \theta_c C_b \left( \sum_j^N h_{(m,j)} \delta \lambda_{(m,j)}^c e^{-TTL^c \sum_j^N \lambda_{(m,j)}^c} - \sum_j^N h_{(m,j)} \lambda_{(m,j)}^c \left( \sum_j^N \lambda_{(m,j)}^c e^{-TTL^c \sum_j^N \lambda_{(m,j)}^c} \delta t_c + TTL^c e^{-TTL^c \sum_j^N \lambda_{(m,j)}^c} \sum_h^N \delta \lambda_{(m,h)}^c \right) \right) & \text{if } m = o \end{cases} \quad (14)$$

- *Policy 2*:

$$\delta \mathrm{J}_m^c(TTL^c, \delta t_c, \mathbf{\Lambda}_m^c, \delta \mathbf{\Lambda}_m^c) = \theta_c C_s \left( \frac{\sum_j^N \lambda_{(m,j)}^c}{(1 + TTL^c \sum_j^N \lambda_{(m,j)}^c)^2} \delta t_c + \frac{TTL^c}{(1 + TTL^c \sum_j^N \lambda_{(m,j)}^c)^2} \sum_h^N \delta \lambda_{(m,h)}^c \right)$$
$$+\theta_c C_b \left( \sum_j^N h_{(m,j)} \delta \lambda_{(m,j)}^c \frac{TTL^c \sum_j^N \lambda_{(m,j)}^c}{1 + TTL^c \sum_j^N \lambda_{(m,j)}^c} + \sum_j^N h_{(m,j)} \lambda_{(m,j)}^c \left( \frac{\sum_j^N \lambda_{(m,j)}^c}{(1 + TTL^c \sum_j^N \lambda_{(m,j)}^c)^2} \delta t_c + \frac{TTL^c}{(1 + TTL^c \sum_j^N \lambda_{(m,j)}^c)^2} \sum_h^N \delta \lambda_{(m,h)}^c \right) \right)$$
$$+ \begin{cases} 0 & \text{if } m \neq o \\ \theta_c C_b \left( \sum_j^N h_{(m,j)} \delta \lambda_{(m,j)}^c \frac{1}{1 + TTL^c \sum_j^N \lambda_{(m,j)}^c} - \sum_j^N h_{(m,j)} \lambda_{(m,j)}^c \left( \frac{\sum_j^N \lambda_{(m,j)}^c}{(1 + TTL^c \sum_j^N \lambda_{(m,j)}^c)^2} \delta t_c + \frac{TTL^c}{(1 + TTL^c \sum_j^N \lambda_{(m,j)}^c)^2} \sum_h^N \delta \lambda_{(m,h)}^c \right) \right) & \text{if } m = o \end{cases} \quad (15)$$

where $\delta \Lambda_m^c$ and $\delta \Omega_m^c$ are both $1 \times N$ vectors.

#### 5.2.2 Prediction of Variation of Resource Cost

The differential cost $\delta \mathrm{J}^c$ can be divided into the resouce cost $\delta \mathrm{J}_m^c$ of $CN_m$ as shown in Appendix. The differential cost for the two policies are shown in Eqs. (14) and (15). By substituting $TTL_k^c$, $\mathbf{\Lambda}_m^c$ in Eq. (12), and $\delta \mathbf{\Lambda}_m^c$ in Eq. (13) for these models, $\delta \mathrm{J}_m^c$ can be expressed as a linear function of $\delta t_c$. Furthermore, $CN_m$ informs its parent $CN$ of the cumulative differential cost as follows.

$$\delta \mathrm{J}_m^c + \sum_{h \in \mathrm{L}_m} \delta \mathrm{J}_h^c$$

where $\mathrm{L}_m$ is a set of $CN$s below $CN_m$ on the delivery tree.

Finally, the TTL controller on the root $CN$ can calculate the sum of differential costs $\sum_{m=1}^M \delta \mathrm{J}_m^c$ as $Q_c \delta t_c$ in Eq. (11). As a result, the proposed TTL controller can decide the manipulated value $\delta t_c$ of TTL.

## 6. Evaluation

We evaluate the effectiveness of the proposed mechanism when controlling the TTL value of content based on the predictive models. The evaluation conditions are set to the following.

- **Test network:** 4-layered tree topology with in total 85 nodes of which 64 nodes are edge nodes having requesting users, cf. Fig. 7. The root node has an origin server. The memory size of each $CN$ is set to sufficient size to cache $10^3$ objects.
- **Content information:** Zipf-distributed requests from each site $i$ for $K = 10^3$ objects are defined as $r_i^c = \gamma k^{-\alpha}/c$, $c = \sum_{k=1}^K k^{-\alpha}$, cf. Fig. 8. We set $\alpha$ to 0.8 for User Generated Content (UGC) and 1.2 for VoD [20]–[22] and $\gamma$ to 100 [requests/sec]. For analytical simplicity, we assume that each content object has the same size of $\theta_c$ which is set to 1.
- **Cost parameters:** The cost ratios between *storage cost* $C_s$ and *bandwidth cost* $C_b$ are set to various values as 0.03 : 0.97, 0.1 : 0.9, 0.5 : 0.5, and 0.9 : 0.1. Here, the cost ratio of 0.03 : 0.97 corresponds to the energy cost as shown in Table 2
- **Control parameters:** We assume the manipulated value $\delta t_c$ as 1 sec. Moreover, initial TTLs of content and the maximum TTL are set to 1 sec and 300 sec, respectively.

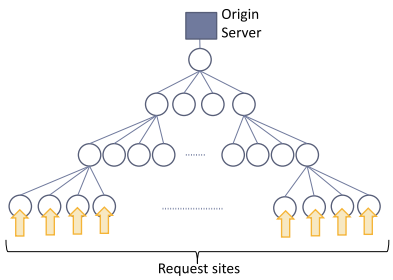



**Fig. 7** Tree topology for evaluation.

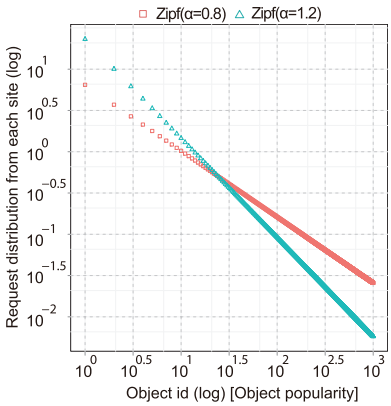



**Fig. 8** Request distribution from each site.

**Table 2** Power cost parameters.

| Device (Product) | Power / Spec | Power Cost per 1 sec |
|---|---|---|
| Memory (DRAM) | 10 W / 4 GB | $C_s = 3.125 \times 10^{-10}$ J/bit |
| Node (CRS-1) | 4185 W / 320 Gbps | $C_b = 1.3 \times 10^{-8}$ J/bit |

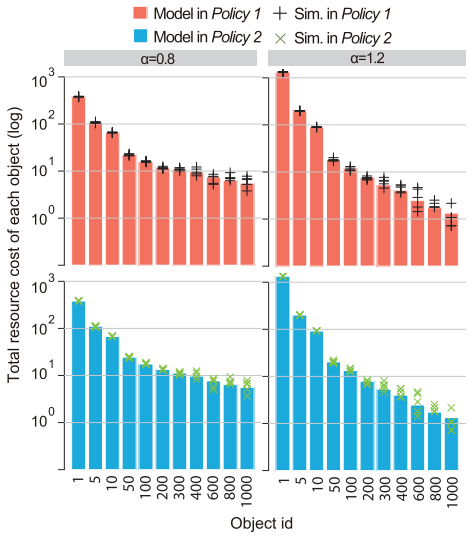
**Table 3** Comparison between numerical evaluation and simulation.

| Process | | Numerical Calculation | Simulation |
|---|---|---|---|
| input | request rate $r_i^c$ | given by a static value | measured by each edge *CN* |
| cache system | cache probability | calculated by Eqs. (1), (2) and controlled TTLs | measured by each *CN* |
| | request propagation and data delivery | calculated by Eqs. (3) and (6) | simulated by packet process |
| output | total cost $J^c$ | calculated by Eq. (5) and controlled TTLs | measured by each *CN* |

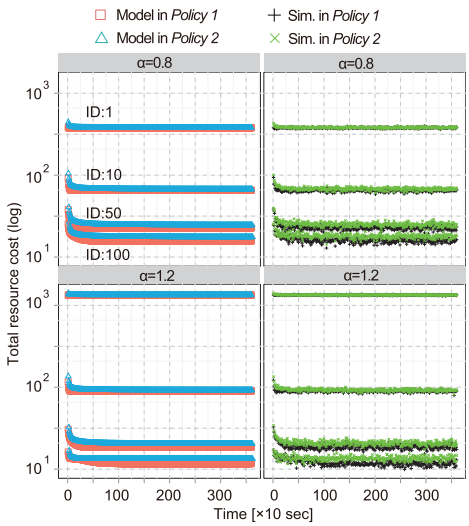
## 6.1 Comparison of the Proposed Model and Simulation

To verify the validity of the model proposed in Sect. 4, we first compare control results calculated by the proposed model with that measured by simulations for 11 representative objects with $\alpha = 0.8$ and $\alpha = 1.2$ in $C_s : C_b = 0.1 : 0.9$.

Here, we show the difference between the numerical calculation based on the proposed model and the packet simulation in Table 3. Both evaluations are executed by the distributed mechanism in Sect. 5.2 and TTL control in Eq. (11). Meanwhile in the numerical evaluation, the request rate $r_i^c$ as input is set to a static value and the total cost $J^c$ as output is calculated by Eq. (5) and the controlled TTL value. In the simulation, the request rate $r_i^c$ is measured as the number of requests by each edge *CN* and the total cost $J^c$ is calculated as the sum of *storage cost* and *bandwidth cost* measured by each *CN*. Furthermore, the request propagation and data delivery are simulated as packet processes and we set the control interval $\Delta$ and simulation time as 10 sec and 3600 sec, respectively. Moreover to measure the average rate of user requests $r_j$ within 10 sec in the same way as [19], we use




(a) Comparison of controlled cost




(b) Time change of cost for object ids: 1, 10, 50, and 100

**Fig. 9** Total resource cost calculated by the proposed model and measured by simulations for different object ids in $C_s : C_b = 0.1 : 0.9$.

EWMA in consideration of historical data in the past 600 sec.

In Fig. 9 (a), we show the total resource cost estimated by the model and the average cost within 10 sec measured from 3560 sec to 3600 sec in the simulation. Additionally, Fig. 9 (b) shows the time change of total resource cost estimated by the proposed model and measured by simulation for 4 objects. The average cost measured by simulation varies more widely for content objects having lower request rates but the results estimated by the proposed model provide suitable approximations of the simulation results. Therefore, we see that the proposed model can express the statistical characteristics for the proposed control mecha-

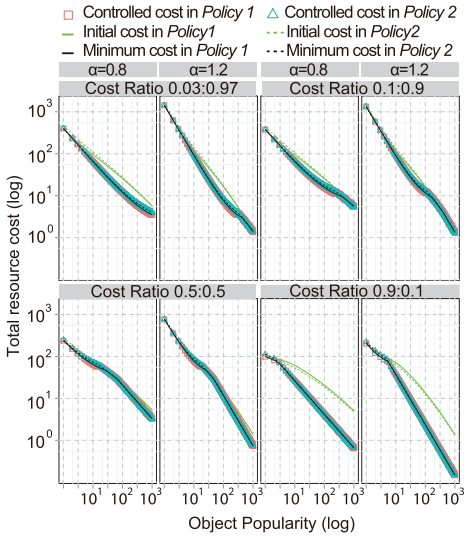



**Fig. 10** Convergence value of resource cost for content objects.

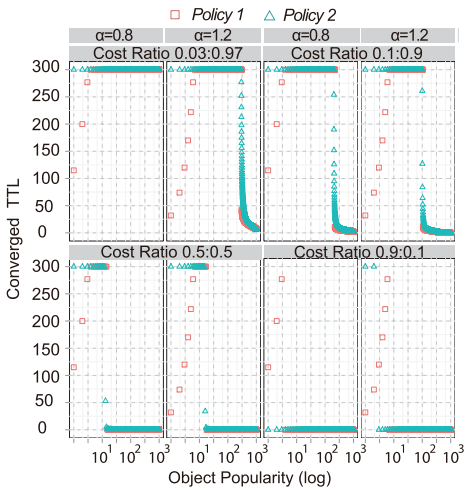



**Fig. 11** Converged TTLs for content objects.

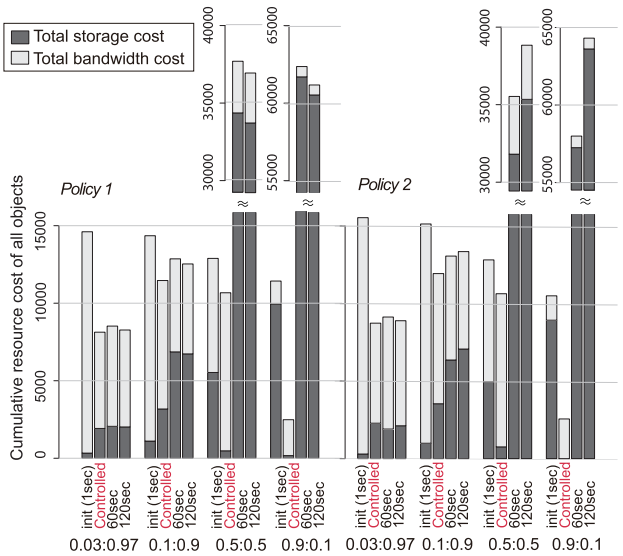
nism.

## 6.2 Effectiveness of the Proposed Mechanism

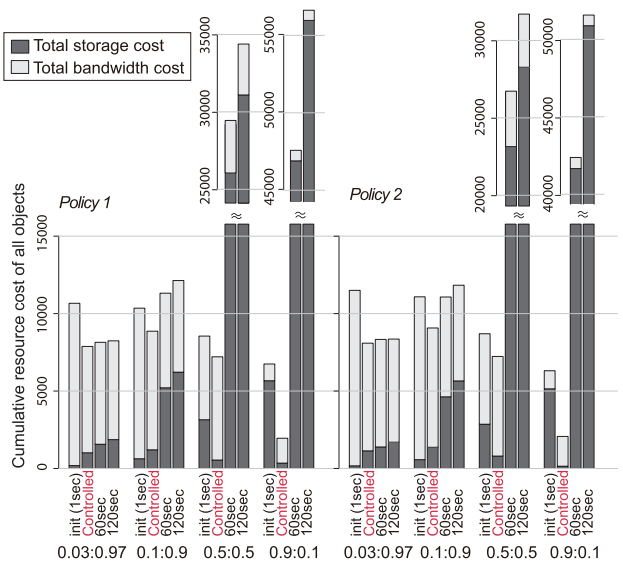
Next, we demonstrate the effectiveness of the proposed mechanism by using the model-based analysis.

Figure 10 shows the total resource cost for each object when setting the initial TTLs to 1 sec, the controlled cost by the proposed mechanism, and the minimum cost in the given range of TTLs from 1 sec to 300 sec which are calculated offline by the request propagation model in Eq. (3), respectively. These results demonstrate that the proposed mechanism is able to find TTL values that reduce the resource cost to near the minimum cost under the given conditions, i.e., the control range of TTL values, initial TTL values, the policy of TTL-based caching, and cost parameters.

Figure 11 shows the converged TTLs controlled by the proposed mechanism. As a result, the controlled TTLs become smaller as *storage cost* becomes larger. Moreover, the




(a) Objects with $\alpha = 0.8$




(b) Objects with $\alpha = 1.2$

**Fig. 12** Cumulative resource cost of all objects and the effectiveness of reducing the total resource cost (initial cost, controlled cost, cost when setting TTLs to 60 and 120 sec).

TTL values of content having large request rates converge at large values which are saturated by the maximum TTL of 300 sec. Here, the TTL values in *Policy 1* from id 1 to id 10 are not saturated at 300 sec in spite of high popularity objects. This why the cache probability of Eq. (1) for these objects in *Policy 1* becomes 1 even when the TTL values are small.

Additionally, Fig. 12 presents the cumulative resource cost of all objects. To compare the effectiveness of the proposed method with conventional methods using static TTLs, these figures additionally show the results when set-

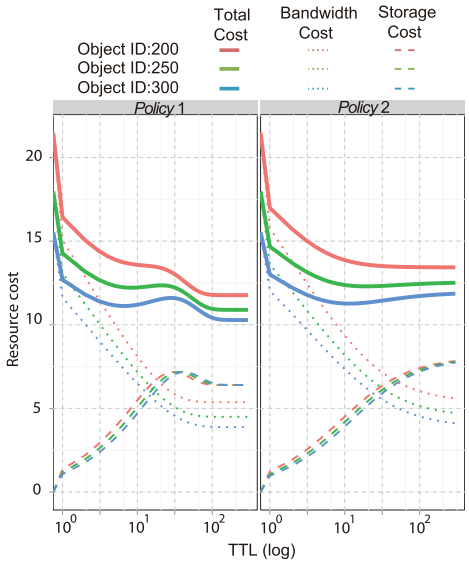



**Fig. 13** Tradeoff between storage cost and bandwidth cost for object ids: {200, 250, 300} with $\alpha = 0.8$ in $C_s : C_b = 0.1 : 0.9$.

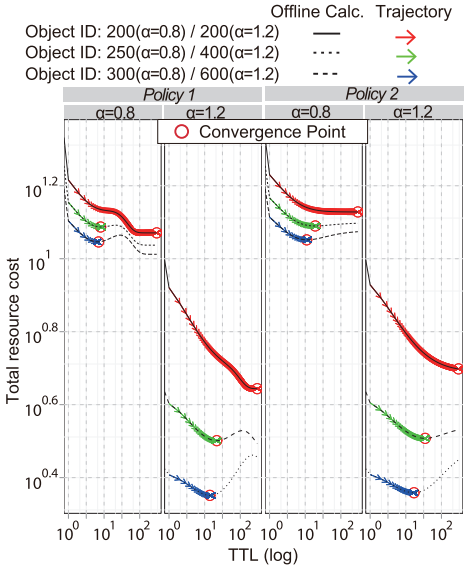



**Fig. 14** Control trajectory for objects with $\alpha = 0.8$ in $C_s : C_b = 0.1 : 0.9$ and with $\alpha = 1.2$ in $C_s : C_b = 0.03 : 0.97$ when the initial TTL is set to 1 sec.

ting static TTLs of all objects to 1 sec, 60 sec, and 120 sec.

In these results, we see that the proposed mechanism can reduce the total resource cost compared to the cases when setting the TTL values of all objects to 1 sec as the initial value. Furthermore, this mechanism can locally search for TTL values which can realize lower cost than that of when the TTL values are set to static values of 60 sec and 120 sec.

### 6.3 Verification of Optimality

Figure 13 presents the tradeoffs between storage cost and bandwidth cost for object ids: {200, 250, 300} with $\alpha = 0.8$ in $C_s : C_b = 0.1 : 0.9$, which are derived by the offline calculation using the request propagation model in Eq. (3). These results show that the total storage cost of all *CN*s in *Policy 1* is not an increasing monotonic function. This is why there is a tradeoff between the total storage cost and TTL in *Policy 1*, which means that the cache probability of the other nodes except for edge nodes approaches 0 as the cache probability of edge nodes is close to 1. As a result, it is conceivable that the curve of total resource cost depends on the request distribution and target topology.

Figure 14 presents the control trajectories of the proposed mechanism and the minimum cost derived by offline calculation for object ids: {200, 250, 300} with $\alpha = 0.8$ in $C_s : C_b = 0.1 : 0.9$ and object ids: {200, 400, 600} with $\alpha = 1.2$ in $C_s : C_b = 0.03 : 0.97$. Additionally in Fig. 15, we present the change of TTL values of each content. In these results, the proposed controllers in *Policy 1* converge at the TTL value and drop to the local minimum cost. Meanwhile, the TTL controllers in *Policy 2* can search for the minimum cost in the control range of TTLs by descending the slope of the cost function.

In Fig. 16, we change the initial TTLs from 1 sec to 150

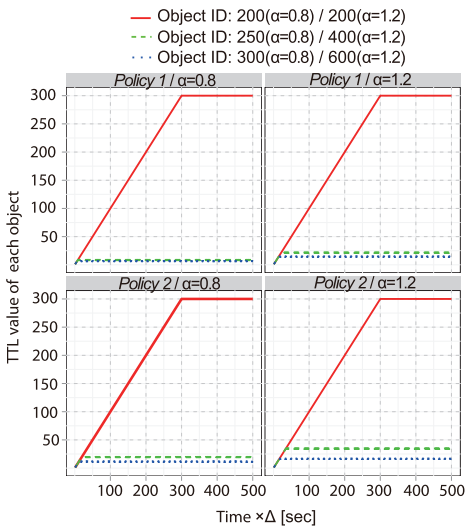



**Fig. 15** Time change of TTLs for objects with $\alpha = 0.8$ in $C_s : C_b = 0.1 : 0.9$ and with $\alpha = 1.2$ in $C_s : C_b = 0.03 : 0.97$ when the initial TTL is set to 1 sec.

sec. In the results for object ids: {250, 300} with $\alpha = 0.8$ and object id: {400} with $\alpha = 1.2$ in *Policy 1*, the TTL values converge at different points from those when the initial TTLs are 1. However, because the request rates from users vary momentarily, we can guess that there is little probability of staying on the local minimum cost. Furthermore, in order to avoid converging to local solutions, it is effective to periodically reset the TTL values randomly.

### 6.4 Impact of the TTL Control on Cache Performance

We show the cache miss ratio and average hop length in the control results derived by Fig. 10. The cache miss ra-

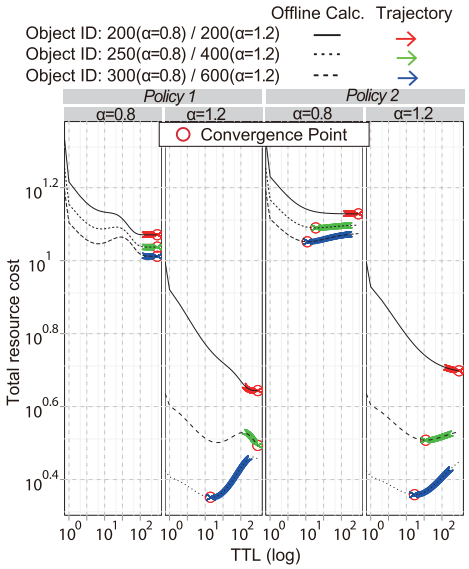



**Fig. 16** Control trajectory for object with $\alpha = 0.8$ in $C_s : C_b = 0.1 : 0.9$ and with $\alpha = 1.2$ in $C_s : C_b = 0.03 : 0.97$ when the initial TTL is set to 150 sec.

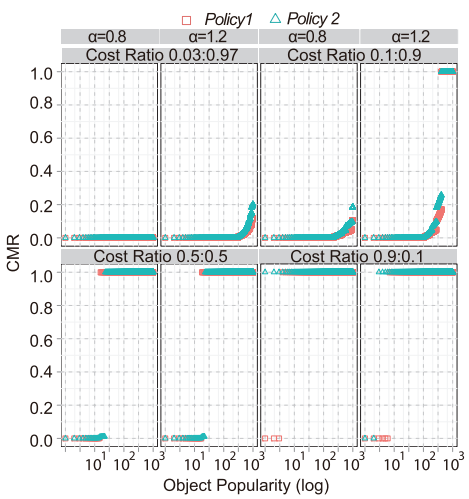



**Fig. 17** Cache miss ratio of all objects in the network when converging at the minimum cost.

tio $\mathrm{CMR}^c$ and the average hop length $\mathrm{AHL}^c$ of object $c$ in the network can be defined as the following metrics.

$$\mathrm{CMR}^c := \frac{\sum_j^N \lambda_{(o,j)}^c \left(1 - f(\sum_k^N \lambda_{(o,k)}^c, TTL^c)\right)}{\sum_j^N r_j^c} \tag{16}$$

$$\mathrm{AHL}^c := \frac{\sum_j^N Tr_j^c}{\sum_j^N r_j^c} \tag{17}$$

Figures 17 and 18 show the cache miss ratio and the average hop length for all contents when the proposed controller converges at each equilibrium. In the results, as content has lower request rates and *storage cost* becomes larger, the cache miss ratio and average hop length of content become higher and longer, respectively.

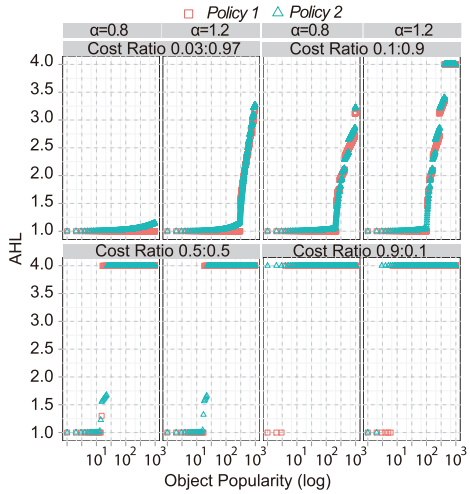



**Fig. 18** Average hop length of all objects when converging at the minimum cost.

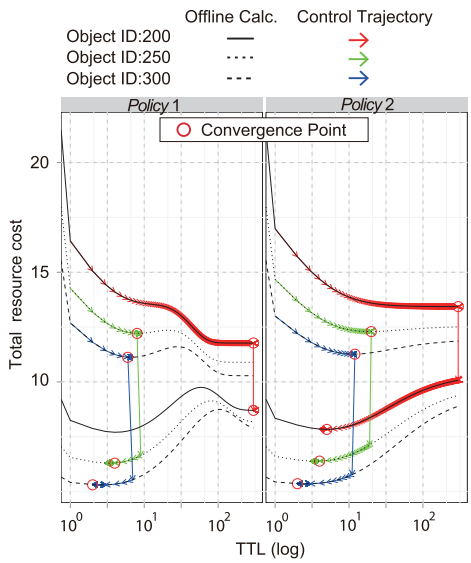



**Fig. 19** Control trajectory for object ids: {200, 250, 300} in $C_s : C_b = 0.1 : 0.9$.

## 6.5 Adaptability of TTL Control

We finally demonstrate the adaptability of the proposed mechanism. Figure 19 presents the control trajectory for object ids: {200, 250, 300} in $C_s : C_b = 0.1 : 0.9$ when the request rate is changed from $\alpha = 0.8$ to $\alpha = 1.2$ in the middle of the cache operation. In these results, the proposed controller tracks the curve of the cost function caused by the change of the request rate of content. Meanwhile for object ids: {250, 300}, the controllers in *Policy 1* first converge at the local minimum cost for the cost curve with $\alpha = 0.8$ and can search for the minimum cost in the control range of TTLs for the cost curve with $\alpha = 1.2$ after changing the request rate of content. Therefore, we see that our proposed mechanism can adaptively search for a TTL value to reduce

the total resource cost in the distributed cache system according to the change of the request distribution.

## 7. Conclusions

We proposed an adaptive control mechanism which autotunes the TTL value of content based on predictive models to reduce the total resource cost in distributed cache systems and introduced a distributed solution. The proposed mechanism periodically decides the manipulated values of TTL by predicting the impact of the TTL values on the total resource cost in the hierarchical cache system. In the evaluations, we compared the control results estimated by the proposed model with those obtained by simulations. Furthermore, we analyzed the effectiveness of the proposed mechanism and showed that our proposed mechanism can search for a TTL value of content to reduce the total resource cost by descending the slope of the cost function. Additionally, we analyzed the cache performance in the proposed control mechanism by using performance metrics such as cache miss ratio and average hop length and finally demonstrated the adaptability of the proposed mechanism. As future work, we plan on enhancing the mechanism to avoid converging at local solutions and extending the proposed mechanism to chunk level caching.

### References

[1] G. Xylomenos, C.N. Ververidis, V.A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K.V. Katsaros, and G.C. Polyzos, "A survey of information-centric networking research," IEEE Communications Surveys & Tutorials, pp.1–26, 2013.

[2] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard, "Networking named content," Proc. 5th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2009), pp.1–12, Rome, Italy, Dec. 2009.

[3] A. Fischer, J.F. Botero, M.T. Beck, H. Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," IEEE Communications Surveys & Tutorials, vol.15, no.4, pp.1888–1906, 2013.

[4] Y, Chen, J. Li, T. Wo, C. Hu, and W. Liu, "Resilient virtual network service provision in network virtualization environments," IEEE 16th International Conference on Parallel and Distributed Systems (ICPADS), pp.51–58, Dec. 2010.

[5] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," IEEE J. Sel. Areas Commun., vol.20, no.7, pp.1305–1314, Sept. 2002.

[6] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for lru cache performance," Proc. 24th International Teletraffic Congress (ITC), pp.1–8, Krakow, Poland, Sept. 2012.

[7] D.S. Berger, P. Gland, S. Singla, and F. Ciucu, "Exact analysis of ttl cache networks: The case of caching policies driven by stopping times," CoRR, vol.abs/1402.5987, 2014.

[8] C.T. Kelley, Iterative Methods for Optimization, Society for Industrial and Applied Mathematics, 1999.

[9] W. Li, E. Chan, Y. Wang, D. Chen, and S. Lu, "Cache placement optimization in hierarchical networks: Analysis and performance evaluation," 7th International IFIP-TC6 Networking Conference, pp.385–396, Singapore, May 2008.

[10] S. Borst and V. Gupta, "Distributed caching algorithms for content distribution networks," Proc. 29th IEEE Conference on Computer Communications (INFOCOM 2010), pp.1478–1486, San Diego, CA, USA, March 2010.

[11] G. Carofiglio, V. Gehlen, and D. Perino, "Experimental evaluation of memory management in content-centric networking," Proc. IEEE International Conference on Communications (ICC), pp.1–6, Kyoto, Japan, 2011.

[12] Y.T. Hou, J. Pan, Bo Li, and S.S. Panwar, "On expiration-based hierarchical caching systems," IEEE J. Sel. Areas Commun., vol.22, pp.134–150, Jan. 2004.

[13] Y.C. Fofack, P. Nain, G. Neglia, and D. Towsley, "Performance evaluation of hierarchical ttl-based cache networks," Comput. Netw., vol.65, pp.212–231, March 2014.

[14] I. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement in arbitrary networks," Proc. 12th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp.661–670, Washington, DC, USA, Jan. 2001.

[15] L. Qiu, V.N. Padmanabhan, and G.M. Voelker, "On the placement of web server replicas," Proc. 20th IEEE Conference on Computer Communications (INFOCOM 2001), pp.1587–1596, Anchorage, AK, USA, April 2001.

[16] K. Guan, G. Atkinson, and D.C. Kilper, "On the energy efficiency of content delivery architectures," Proc. 4th IEEE International Conference on Communications (ICC) Workshop on Green Communications, Kyoto, Japan, June 2011.

[17] S. Imai, K. Leibnitz, and M. Murata, "Energy efficient data caching for content dissemination networks," J. High Speed Networks, vol.19, no.3, pp.215–235, Oct. 2013.

[18] S. Imai, K. Leibnitz, and M. Murata, "Modeling of content dissemination networks on multiplexed caching hierarchies," The Thirteenth International Conference on Networks, pp.111–118, Nice, Frace, Feb. 2014.

[19] F.N. Santos, B. Ertl, C. Barakat, T. Spyropoulos, and T. Turletti, "CEDO: Content-centric dissemination algorithm for delay-tolerant networks," MSWiM, pp.377–386, Nov. 2013.

[20] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing," 2011.

[21] C. Fricker, P. Robert, and J. Roberts, "Impact of traffic mix on caching performance in a content-centric network," Proc. IEEE NOMEN'12, Workshop on Emerging Design Choices in Name-Oriented Networking, pp.310–315, Orlando, Florida, USA, March 2012.

[22] P. Gill, M. Arlitt, and Z. Li, "Youtube traffic characterization: A view from the edge," Proc. IMC'07, pp.15–28, San Diego, CA, USA, Oct. 2007.

## Appendix: Differentiation of Total Resource Cost

We show the differentiation of the total resource cost function $\mathrm{J}^c(TTL^c, \delta t_c, \mathbf{\Lambda}^c, \delta\mathbf{\Lambda}^c)$ in Eq. (A· 1).

$$\delta \mathrm{J}^c = \frac{\partial \mathrm{J}^c}{\partial TTL}\delta t_c + \sum_{(m,n)} \frac{\partial \mathrm{J}^c}{\partial \lambda^c_{(m,n)}}\delta\lambda^c_{(m,n)} \tag{A· 1}$$

$$= \theta_c C_s[1,..,1]\left(\begin{bmatrix} \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(1,j)}, TTL^c)}{\partial TTL} \\ \vdots \\ \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(M,j)}, TTL^c)}{\partial TTL} \end{bmatrix}\delta t_c \right.$$

$$\left. + \left( \begin{bmatrix} \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(1,j)}, TTL^c)}{\partial \lambda^c_{(1,1)}} & .. & \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(1,j)}, TTL^c)}{\partial \lambda^c_{(1,N)}} \\ & \vdots & \\ \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(M,j)}, TTL^c)}{\partial \lambda^c_{(M,1)}} & .. & \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(M,j)}, TTL^c)}{\partial \lambda^c_{(M,N)}} \end{bmatrix} * \delta\mathbf{\Lambda}^c \right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right)$$

$$
\begin{aligned}
&+\theta_c C_b\Bigg([1,..,1](\mathbf{Hp}*\delta\mathbf{\Lambda}^c)^{\mathrm{T}}\begin{bmatrix} \boldsymbol{f}(\sum_j^N \lambda^c_{(1,j)}, TTL^c) \\ \vdots \\ \boldsymbol{f}(\sum_j^N \lambda^c_{(M,j)}, TTL^c) \end{bmatrix} \\
&+[1,..,1](\mathbf{Hp}*\mathbf{\Lambda}^c)^{\mathrm{T}}\Bigg(\begin{bmatrix} \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(1,j)}, TTL^c)}{\partial TTL} \\ \vdots \\ \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(M,j)}, TTL^c)}{\partial TTL} \end{bmatrix}\delta t_c \\
&+\Bigg(\begin{bmatrix} \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(1,j)}, TTL^c)}{\partial \lambda^c_{(1,1)}} & .. & \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(1,j)}, TTL^c)}{\partial \lambda^c_{(1,N)}} \\ & \vdots & \\ \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(M,j)}, TTL^c)}{\partial \lambda^c_{(o,1)}} & .. & \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(M,j)}, TTL^c)}{\partial \lambda^c_{(o,N)}} \end{bmatrix} * \delta\mathbf{\Lambda}^c\Bigg)\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}\Bigg)\Bigg) \\
&+(\mathbf{Hp}[o,]*\delta\mathbf{\Lambda}^c[o,])^{\mathrm{T}}(1-\boldsymbol{f}(\sum_j^N \lambda^c_{(o,j)}, TTL^c)) \\
&-(\mathbf{Hp}[o,]*\mathbf{\Lambda}^c[o,])^{\mathrm{T}}\Bigg(\frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(o,j)}, TTL^c)}{\partial TTL}\delta t_c \\
&+\Bigg(\begin{bmatrix} \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(o,j)}, TTL^c)}{\partial \lambda^c_{(1,1)}} & .. & \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(o,j)}, TTL^c)}{\partial \lambda^c_{(1,N)}} \end{bmatrix} \\
&*\delta\mathbf{\Lambda}^c[o,]\Bigg)\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}\Bigg)
\end{aligned}
$$

Furthermore, this differential cost can be divided into the resource cost function $\delta \mathrm{J}^c_m(TTL^c, \delta t_c, \mathbf{\Lambda}^c_m, \delta\mathbf{\Lambda}^c_m)$ of $CN_m$ as follows.

$$
\begin{aligned}
\delta \mathrm{J}^c_m &= \theta_c C_s\Bigg(\frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(m,j)}, TTL^c)}{\partial TTL}\delta t_c \\
&+\sum_h^N \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(m,j)}, TTL^c)}{\partial \lambda^c_{(m,h)}}\delta\lambda^c_{(m,h)}\Bigg) \\
&+\theta_c C_b\Bigg(\sum_j^N h_{(m,j)}\delta\lambda^c_{(m,j)}\boldsymbol{f}(\sum_j^N \lambda^c_{(m,j)}, TTL^c) \\
&+\sum_j^N h_{(m,j)}\lambda^c_{(m,j)}\Bigg(\frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(m,j)}, TTL^c)}{\partial TTL}\delta t_c \\
&+\sum_h^N \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(m,j)}, TTL^c)}{\partial \lambda^c_{(m,h)}}\delta\lambda^c_{(m,h)}\Bigg)\Bigg) \qquad (\mathrm{A}\cdot 2) \\
&+\begin{cases} 0 & \text{if } m \neq o \\ \theta_c C_b\Bigg(\sum_j^N h_{(m,j)}\delta\lambda^c_{(m,j)}\left(1-\boldsymbol{f}(\sum_j^N \lambda^c_{(m,j)}, TTL^c)\right) \\ -\sum_j^N h_{(m,j)}\lambda^c_{(m,j)}\Bigg(\frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(m,j)}, TTL^c)}{\partial TTL}\delta t_c \\ +\sum_h^N \frac{\partial \boldsymbol{f}(\sum_j^N \lambda^c_{(m,j)}, TTL^c)}{\partial \lambda^c_{(m,h)}}\delta\lambda^c_{(m,h)}\Bigg)\Bigg) & \text{if } m = o \end{cases}
\end{aligned}
$$

**Satoshi Imai** received the M.E. degree in systems science from Osaka University, Japan in 2004. In April 2004, he joined Fujitsu Laboratories Ltd., Japan. Since April 2012, he is a doctoral student at the Graduate School of Information Science and Technology, Osaka University. He is a member of IEICE and IEEE. His current research interests include modeling and analysis of communication systems for future networks, especially the methodology for in-network caching and network management on network virtulization.

**Kenji Leibnitz** received his master and Ph.D. degrees in information science from the University of Würzburg in Germany. In May 2004, he joined Osaka University, Japan, as a Postdoctoral Research Fellow and from July 2006 he was a Specially Appointed Associate Professor at the Graduate School of Information Science and Technology. Since April 2010 he is a Senior Researcher at the National Institute of Information and Communications Technology (NICT), as well as a Guest Associate Professor at Osaka University and from April 2013 he is with the Center of Information and Neural Networks (CiNet) of NICT and Osaka University. His research interests are in modeling and performance analysis of communication networks, especially the application of biologically and brain inspired mechanisms to self-organization in future networks.

**Masayuki Murata** received the M.E. and D.E. degrees in Information and Computer Science from Osaka University, Japan, in 1984 and 1988, respectively. In April 1984, he joined Tokyo Research Laboratory, IBM Japan, as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. In April 1999, he became a Professor of Cybermedia Center, Osaka University, and he is now with the Graduate School of Information Science and Technology, Osaka University, since April 2004. He has more than eight hundred papers of international and domestic journals and conferences. His research interests include computer communication network architecture, performance modeling and evaluation. He is a member of IEEE, ACM and IEICE. Also, he is now partly working at NICT (National Institute of Information and Communications Technology) as Deputy of New-Generation Network R&D Strategic Headquarters.