IEICE TRANSACTIONS

on Communications

DOI:10.23919/transcom.2023EBP3152

This advance publication article will be replaced by the finalized version after proofreading.

A PUBLICATION OF THE COMMUNICATIONS SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

PopDCN: Popularity-Aware Dynamic Clustering Scheme for Distributed Caching in ICN*

Mikiya YOSHIDA[†a)], **Yusuke ITO**[††b)], **Yurino SATO**[†††c)], *and* **Hiroyuki KOGA**[††d)], *Members*

SUMMARY Information-centric networking (ICN) provides low-latency content delivery with in-network caching, but delivery latency depends on cache distance from consumers. To reduce delivery latency, a scheme to cluster domains and retain the main popular content in each cluster with a cache distribution range has been proposed, which enables consumers to retrieve content from neighboring clusters/caches. However, when the distribution of content popularity changes, all content caches may not be distributed adequately in a cluster, so consumers cannot retrieve them from nearby caches. We therefore propose a dynamic clustering scheme to adjust the cache distribution range in accordance with the change in content popularity and evaluate the effectiveness of the proposed scheme through simulation.

key words: *ICN, distributed caching, dynamic clustering*

1. Introduction

Information-centric networking (ICN) [3], [4] has been attracting attention as a new architecture that uses network caching to satisfy the requirements (e.g., ultra-low latency, ultra-high reliability, and massive connectivity) for emerging services such as IoT-like automation, robotics, and industrial automation [5], [6]. In ICN, a consumer sends interest packets containing content names to request content. A content router (CR), which is an intermediate router receiving the interest packets, forwards the packets to a producer on the basis of a routing table called a forwarding information base (FIB). The producer then returns data packets of the requested content with the reverse path to consumers. The CRs cache data packets on their content store (CS) during forwarding, so they can return caches to consumers instead of the producer if they store the requested data. Namely, this in-network caching, which can satisfy the future requests of consumers, significantly reduces the network load and improves content delivery efficiency. To take full advantage of in-network caching, an efficient content caching scheme is needed, and various schemes have been proposed.

Simple content caching makes a cache decision on individual CRs, while distributed content caching distributes content by considering nearby caches to satisfy content requests. Distributed caching solves a cache efficiency problem that simple content caching causes cache duplication for a small amount of highly popular content over neighboring CRs. However, if the cached required content is distributed over a large range, delivery latency may increase.

Therefore, cluster-based distributed caching schemes have been proposed [7], [8] to control a distributed range. These schemes group CRs into clusters in a domain[†] and retain the main popular content in each cluster using a distributed caching manner. This aims to avoid cache duplication among CRs in the cluster, enabling the caches of each content to be distributed within it. As a result, the delivery latency can be controlled by cluster size, and it enables consumers to retrieve content efficiently from the originating clusters. However, a too-small distribution range decreases cache utilization and causes delivery delays due to the delivery from producers, while a too-large distribution range increases cache utilization but may cause delivery delays due to long cache delivery. Therefore, we believe that the adequate cache distribution range should be determined in accordance with content popularity on the basis of such trade-off factors. In a practical environment, the distribution of content popularity changes over time, so it is necessary to determine the distribution range depending on the situation.

We therefore propose a dynamic clustering scheme to adjust the cache distribution range, i.e., cluster size, in accordance with the change in content popularity, considering cache utilization and delivery latency. Our scheme controls the cluster size effectively using a simple threshold-based algorithm based on the number of cache updates on CRs in a cluster. Moreover, we evaluate the effectiveness of the proposed scheme compared with conventional schemes through simulation in a situation where content popularity changes.

The main contributions of this paper, updated from previous papers [1], [2], are as follows:

- We discuss recent studies that utilize clustering techniques in ICN.
- We evaluate the proposed scheme compared with conventional schemes in detail and discuss the effective

[†]The author is with the Center for Information Technology and Management, Okayama University, Okayama-shi, 700-8530 Japan.

[††]The authors are with the Graduate School of Environmental Engineering, The University of Kitakyushu, Kitakyushu-shi, 808-0135 Japan.

[†††]The author is with the Department of Control Engineering, National Institute of Technology (KOSEN), Sasebo college, Sasebo-shi, 857-1193 Japan.

*Earlier version of this paper was presented at ACM ICN2022 and APSIPA ASC2022 [1], [2].

a) E-mail: m-yoshida@okayama-u.ac.jp
b) E-mail: y-ito@kitakyu-u.ac.jp
c) E-mail: y-sato@sasebo.ac.jp
d) E-mail: h.koga@kitakyu-u.ac.jp

[†]In this paper, the term 'domain' refers to a large-scale network consisting of one or more ISPs.

threshold settings.
- We investigate the effectiveness of the proposed scheme in practical network topologies.

The rest of this paper is organized as follows. In Section 2, we describe our motivation for this study through a discussion of related works. In Section 3, we describe our scheme. In Section 4, we describe the simulation model and evaluation details. In Section 5, we evaluate the performance of our scheme in comparison with conventional schemes. Finally, in Section 6, we summarize our findings and conclude the paper.

2. Related Work

In this section, we describe an issue of this study through discussions of various content caching schemes to improve content delivery efficiency.

2.1 Distributed Caching

Simple caching schemes such as LCE [3] and Prob(p) [9] make a cache decision on individual CRs. This may cause cache duplication for a few high-popular contents over neighboring CRs because more frequently requested content is likely to be cached. This becomes useless for other content requests. Therefore, distributed caching schemes have been proposed such as MCD [10], WAVE [11], MuNCC [12], and Hash-routing [13], which distribute various content considering nearby caches to satisfy various content requests. The key idea of MCD and WAVE schemes is that each CR moves requested caches to downstream CRs. Namely, the CR caching the requested content sends its cache to the downstream CR and removes it from itself, so that each cache is not duplicated on the default path, i.e., the shortest path to consumers. However, it is unable to avoid cache duplication among neighboring CRs outside the default path.

In contrast, in MuNCC and proposed in [14] schemes, each CR shares cache summaries that are formed using a Bloom filter among neighboring CRs to avoid cache duplication. When a data packet arrives, a CR determines whether it caches it or not depending on the cache summaries of neighboring CRs. The Hash-routing scheme distributes content to CRs using a hash function that maps content identifiers to each CR of the domain, without additional functionality such as shared cache summaries. In particular, when an edge router in the domain receives a request, it calculates the hash value from the received content identifier and forwards it to the responsible CR. Similarly, each CR caches the responsible content whose hash value matches its identifier during forwarding. As a result of this approach, since the cache location of each content is limited to one CR over a domain, it can avoid cache duplication among CRs. However, if the cached required content is distributed over a large range, delivery latency may increase.

2.2 Cluster-Based Distributed Caching

To control the cache distribution range considering delivery latency, network clustering-based distributed caching schemes for ICN have been proposed [7], [8], [15], [16]. These schemes group CRs into clusters in a domain and retain the main popular content in each cluster using a Hash-routing-like distributed caching manner. The delivery latency can thus be controlled by cluster size, enabling consumers to retrieve content efficiently from the originating clusters. As a scheme similar to the aforementioned ones, the HCC [17] scheme has also been proposed. It centrally manages the distributed caches by a cluster header constructed in each cluster. The cluster header calculates the content popularity and importance of each node on the basis of information collected from the cluster, and then assigns the more popular content to the more important node to improve cache efficiency and delivery latency.

However, the amount of content that can be cached in the cluster depends on the cluster size. In other words, a smaller cluster size is insufficient to reduce delivery latency since it cannot cache necessary content sufficiently in the cluster. As mentioned before, in this study, we believe that it is necessary to determine the adequate cache distribution range in accordance with content popularity on the basis of the following trade-off factors. A too-small cache distribution range against the amount of main popular content will not retain sufficient caches, so it decreases cache utilization and causes delivery delays due to the delivery from producers. A too-large cache distribution range can satisfy most user requests within the cluster but causes delivery delays due to the delivery from widely distributed caches. In a practical environment, the distribution of content popularity, i.e., the amount of main popular content, will change over time [18], so it is necessary to determine the distribution range adequately depending on the situation.

3. Proposed Scheme

We propose a dynamic clustering scheme to adjust the cache distribution range in accordance with the change in content popularity. This scheme is an extended version of our previous work [7] that formed a fixed size of clusters. In this section, we explain the operation of the proposed scheme. We first explain the main points of the previous work in Section 3.1, and then explain the extension in detail in Section 3.2.

3.1 Cluster-Based Cache Distribution Scheme

To improve delivery latency and cache efficiency, we have proposed the cluster-based cache distribution scheme. It groups CRs into clusters in a domain and retains the main popular content in each cluster using a distributed caching manner, enabling consumers to retrieve content from the originating clusters. Furthermore, it can also retrieve caches

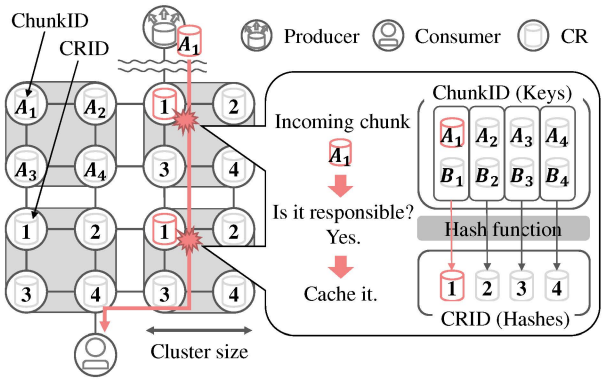


Fig. 1: Cache placement

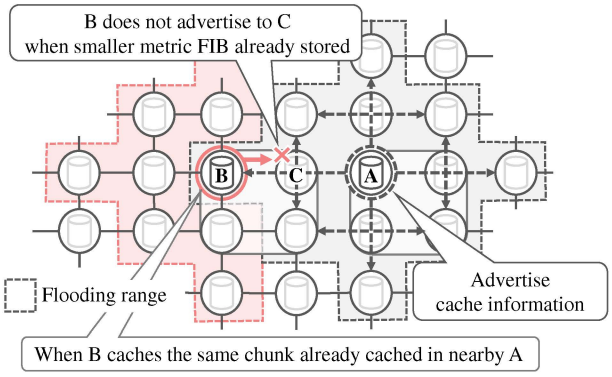


Fig. 2: Cache information advertisement

from closer CRs by advertising cache information among CRs. In the following, we explain two functions of cluster-based distributed caching and advertisement-based routing.

The distributed caching approach uniformly distributes chunks of individual content to all CRs in each cluster, as shown in Fig. 1. This approach improves cache efficiency by avoiding cache duplication in the cluster, leading to more cached content in it. Furthermore, transmission efficiency can also be improved by multi-path cache delivery from multiple CRs (i.e., load balancing).

To uniformly distribute chunks in a cluster, this scheme partitions a domain into clusters of the same size and assigns unique identifiers (CRIDs) to each CR in advance. To avoid cache redundancy among CRs in a cluster, it uses a hash function that maps chunk identifiers to CRIDs. Specifically, when a CR receives a chunk, it caches it as a responsible one if the hash value calculated from the received chunk identifier matches its own CRID. The Least Recently Used (LRU) cache replacement algorithm is used for spaces on the CS. In this study, each cluster is assumed to be a square shape. The cluster size is defined as the number of CRs on one side (as shown in Fig. 1 is 2), which affects the cache efficiency and distance from consumers. Note that the shape of clusters is not important because the main popular content will be retained in clusters if CRIDs are properly assigned within clusters in any topologies. For example, it can be accomplished by defining the cost as the distance between a CR and the nearest CR with a different CRID and solving the problem of minimizing the total cost. This will ensure that each CRID is assigned almost uniformly without bias according to the cluster size, i.e., the number of CRIDs to be assigned. Since each CR is neighboring to CRs with a CRID different from its own, each cluster is nearly a circle shape.

Even if caches are uniformly distributed within a cluster, consumers may not efficiently retrieve all chunks of the requested content from the originating cluster. This is because not all chunks will be cached due to the limitation of total cache capacity in a cluster, or there may be caches on closer CRs in neighboring clusters than those in the originating cluster. Therefore, requests should be forwarded to the nearest caches even those not in the originating clusters regardless of cluster boundaries for efficient content delivery, so the advertisement-based routing approach is used, which forwards interest packets to nearby caches on the basis of the advertised cache information.

To achieve this behavior, each CR informs neighboring CRs of their own responsible cache status. Specifically, CRs that newly cache or discard responsible chunks advertise the cache information (newly cached/discarded) in the flooding manner shown in Fig. 2. The CR receiving the advertised packet updates its FIB entry with the received cache information. Considering the overhead of this operation, the flooding range should be limited but would affect the content retrieval efficiency, which is defined as the flooding limit parameter (as shown in Fig. 2 is 2). This operation is performed only when responsible chunks are cached or discarded, thereby reducing the overhead compared with conventional schemes flooded for all cached chunks such as proposed in [19]. Moreover, to reduce the load caused by flooding, our scheme simply discards and does not forward the flooding packets when it can be determined that neighboring CRs do not need to update their FIB. Let us explain this process using the example shown in Fig. 2. When CR A caches responsible chunks, it advertises its cache information to neighboring CRs (gray-colored range). After that, when CR B caches the same chunk, it can decide not to flood to CR C and advertises the cache information to neighboring CRs except it (red-colored range). This is because CR B has an FIB entry with metric of 2 hops for the chunk by advertised information from CR C and it indicates that CR C already has a valid metric of 1 hop that does not need updating. Namely, if the CRs already have FIB entries of plus 2 hops or fewer metrics than the flooding one, it does not need to advertise it in that direction. Note that this scheme increases overheads including cache information sharing and FIB entry increases to improve acquisition efficiency compared to on-path routing schemes as an inherent issue of off-path routing schemes. To resolve this issue (overheads caused by off-path extension), several solutions (e.g., a Bloom filter approach [12], [14], [20]) have been proposed, while we focus on reducing delivery latency by adjusting cache distribution range while considering only communication overheads caused by flooding in this study so that we will leave this issue for future work.

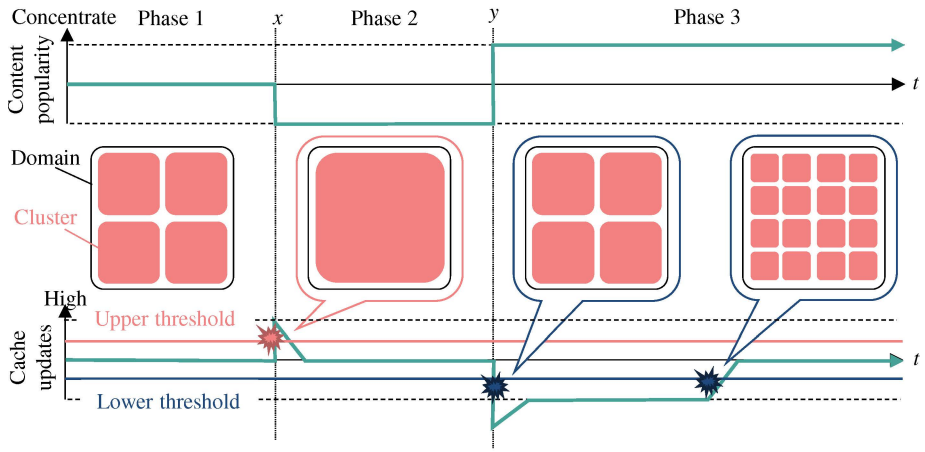


Fig. 3: Operation of dynamic clustering

3.2 Popularity-Aware Dynamic Clustering Scheme

As previously mentioned, the cluster size, i.e., cache distribution range, should be adequately determined in accordance with content popularity. In a practical environment, the distribution of content popularity changes over time, so it is necessary to determine the distribution range depending on the situation. We therefore propose a dynamic clustering scheme to adjust the cache distribution range in accordance with the change in content popularity, considering cache utilization and delivery latency. Our scheme controls the cluster size effectively using a simple threshold-based algorithm based on the number of cache updates in the cluster.

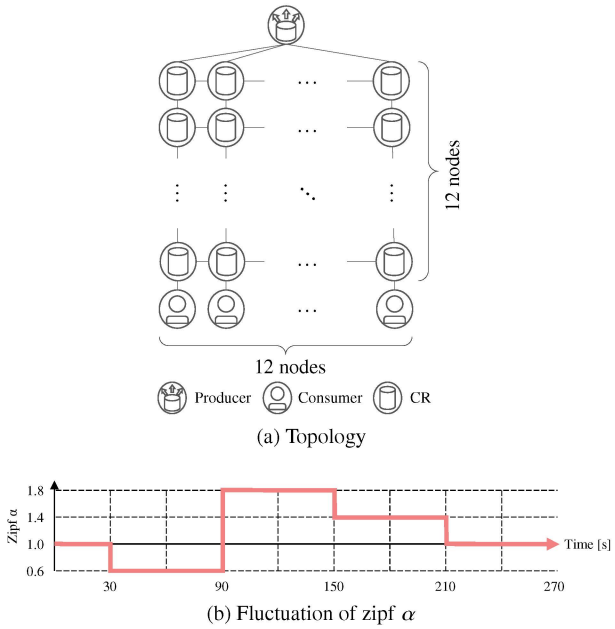
To discuss the adequate cluster size, we focus on the frequency of cache updates in a cluster. This is because this metric is useful to estimate whether the current cluster size is suitable to cache the main popular content. When the frequency of cache updates is high, it indicates that caches are updated by incoming data packets from outside the cluster. Namely, requested content cannot be retrieved inside the cluster as well as the cluster size is too small. A low frequency of cache updates indicates that caches are not updated since requested content can be retrieved inside the cluster. Namely, the cluster size may be decreased to reduce delivery latency. Thus, we consider that the frequency of cache updates in a cluster would fall into a certain range with the appropriate cluster size.

From the aforementioned strategy, the proposed scheme adjusts the cluster size using a simple threshold-based algorithm based on the frequency of cache updates. Specifically, it uses the number of cache updates in a cluster as a metric, and decreases/increases the cluster size when the metric falls below or exceeds lower/upper thresholds. Figure 3 explains how the proposed scheme migrates to the adequate cluster size in accordance with the change in content popularity. Let us consider a t-second scenario when the content popularity will disperse after x seconds, and then heavily concentrate after y seconds. In phase 1 until x seconds, we assume that each cluster, which represents the domain divided into four parts, can store most of the popular content, so the frequency of cache updates fits between the upper and lower thresholds. Namely, the current cluster size is adequate. In phase 2 from x to y seconds when the content popularity disperses, the frequency of cache updates increases and exceeds the upper threshold because the current cluster size cannot retain the popular content sufficiently. Therefore, the cluster size is increased by one level to store them, and therefore the frequency of cache updates decreases and falls within the upper and lower thresholds. In phase 3 after y seconds when the content popularity is heavily concentrated, the cache update frequency decreases and falls below the lower threshold because the current large cluster size has exceeded the sufficient cache capacity compared with the amount of main popular content. Therefore, it attempts to improve the delivery latency by decreasing the cluster size by one level. However, this cluster size still has an excessive cache capacity, so the frequency of cache updates remains below the lower threshold. Therefore, the cluster size is decreased by one more level, and therefore the frequency of cache updates increases and falls within the upper and lower thresholds. Through these procedures, the cluster size can be migrated to the adequate cluster size in accordance with the change in content popularity.

To achieve this function, we assume that a controller is located in a domain and each CR notifies the controller with the number of cache updates. The controller calculates the total number of cache updates separately in each cluster by the information received from each CR. When at least one of the calculated values falls below or exceeds lower/upper thresholds, it reassigns a new CRID and hash function to each CR to decrease/increase cluster size. The cluster size is not changed for a certain period, which is defined as the reclustering interval parameter, immediately after reclustering to mitigate the effect of the heavy fluctuation of cache updates. We believe that such information sharing between the controller and CRs can be achieved by a mechanism like software defined networking (SDN) and the detailed design of the scheme will be left as future work.

Table 1: Simulation parameters

Link bandwidth	1 [Gb/s]
Propagation delay time	5 [ms]
Communication protocol	UDP
Amount of Contents	128
Content size	64 [chunk]
Chunk size	1000 [Byte]
CS size on CR	128 [chunk]
Zipf α	0.6, 1.0, 1.4, 1.8
Cluster size	2, 3, 4, 6, 12
Upper threshold	70–280
Lower threshold	10–150
Reclustering interval	1–24 [s]



(a) Topology

(b) Fluctuation of zipf α

Fig. 4: Simulation model

4. Simulation Model

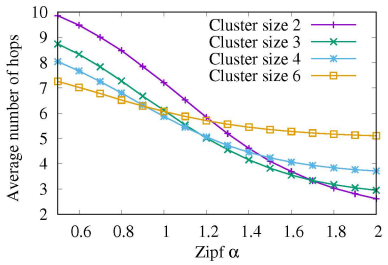
We evaluated the proposed scheme focusing on the effectiveness of retrieving content from nearby clusters/caches in a large domain environment where content popularity changes through simulations using Network Simulator ns-3 ver. 3.30.1 [21] with the implementation of our scheme. We used a simple grid topology with multiple paths to eliminate the effects of cluster shape and content cache placement within clusters as shown in Fig. 4(a) to enable us to focus on the essential effect of dynamically changing cluster size. One producer and 12 consumers were located on the upper and lower sides of the grid (12 x 12) of CRs, respectively. The parameters used in the simulation are summarized in Table 1. The default path is the shortest path to the producer (13 hops from each consumer) and it was set to the FIB of each CR. The ratio of the CS size on CR to the amount of content was set to approximately 1.5% on the basis of comparative papers [8], [22]. The flooding limit was set to 6, which was the best value in terms of cost performance between overhead and efficiency in a preliminary evaluation. As mentioned before, the proposed scheme needs to share information among CRs via the controller, which can be achieved by a number of mechanisms like SDN, and we ignore its effect in this simulation since the exchange of shared information is very infrequent and small compared with data delivery. Each CR notifies the controller of the number of cache updates at 1 second intervals.

Each consumer sent interest packets to request content toward the producer at normal distribution intervals with an average value of 0.3 seconds. The requested content was determined on the basis of the content popularity, in which P2P content was generally known to follow a Zipf-mandelbrot distribution [23]. In this distribution, the degree of bias depends on the parameters α and q. α is the skewness factor that controls the slope of the curve, while $q(\geq 0)$ is known as the plateau factor that determines the flatness of the curve. In this simulation, we gave q a fixed value of 5 and changed the content popularity with α to avoid the complexity of the discussion. Furthermore, we assumed no packet loss occurs so we can focus on the fundamental characteristics of the dynamic clustering approach. The simulation was performed for 270 seconds. We set the Zipf parameter α to 1.0 at the start of the simulation as shown in Fig. 4(b). α changed to 0.6 at 30 seconds after the simulation started, in which a wider range of content is requested, to 1.8 at 90 seconds, to concentrate on the requested content, and after that, it decreases by 0.4 every 60 seconds back to 1.0.

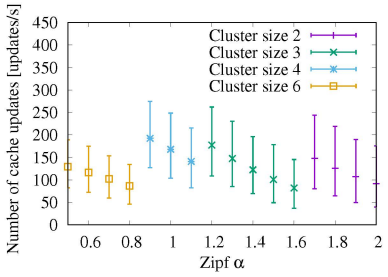
In this simulation, we compared and evaluated the effectiveness of five representative schemes: LCE (LRU), Hash-routing [13], Hash-routing + cluster [8], conventional (Static) [7], and proposed (Dynamic). Furthermore, the average number of hops needed to retrieve content, cache hit rate, and advertisement rate were used as evaluation indices to discuss the effectiveness of our scheme. Note that the Hash-routing + cluster scheme uses the k-split algorithm with the number of hops as similarity metrics for clustering and forms k clusters. The average number of hops focused on content retrieval time, which was defined as the total number of hops during the time when all consumers retrieved content divided by the total number of requests for all consumers. The cache hit rate focused on cache efficiency, which was defined as the total number of cache hits on all CRs divided by the total number of requests for all consumers. The advertisement rate focused on communication overhead, which was defined as the amount of advertisement packets divided by the total amount of traffic. In this study, we assumed the average name length is 30 bytes, and the size of the advertisement packet which includes the content name, the flooding limit, and the flag bit that indicates the cache information (newly cached/discarded), is the same as the Interest packet.

5. Simulation Results

In this section, we first show the effectiveness of our scheme compared with the conventional schemes. Then, we investigate how each parameter including the lower/upper thresh-



(a) Average number of hops



(b) Number of cache updates

Fig. 5: Estimation of adequate thresholds

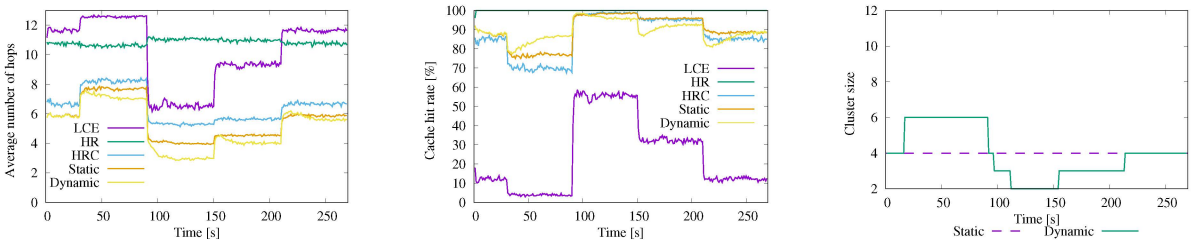
olds and reclustering interval affects our scheme. Finally, we investigate the effect of the change interval of Zipf α and network topology to reveal the environmental tolerance and practicality of our scheme.

5.1 Evaluation of Effectiveness Based on Estimation of Adequate Thresholds

In this section, we first discuss the basis for determining threshold values of the proposed scheme through quantitative evaluations and estimate the effective lower/upper threshold values, which is a key point of the proposed scheme. As mentioned in Section 3.2, given an adequate cluster size, the number of cache updates in the cluster falls into a certain range. We believe that the adequate cluster size can be determined in accordance with the distribution of content popularity. Figure 5 shows the average number of hops and cache updates in the cluster when α varies from 0.5 to 2.0. From Fig. 5(a), we can see that the adequate cluster size is 6 when α is less than 0.9, 4 for α of 1.0–1.1, 3 for α of 1.2–1.6, and 2 for α of 1.7 or larger, respectively, since these cluster size achieve the smallest number of hops for each content popularity. Correspondingly, the number of cache updates in the cluster falls into a certain range when the adequate cluster size is given as shown in Fig. 5(b). Specifically, it is approximately 50 or more for the adequate cluster size of 6 (α = 0.9 or less), 90–280 for the size of 4 (α = 1.0–1.1), 50–270 for the size of 3 (α = 1.2–1.6), and 250 or less for the size of 2 (α = 1.7 or above), respectively. From the aforementioned results, if the number of cache updates in the cluster is approximately 50 and more or 280 and less, the given cluster size will be adequate. Namely, the lower/upper threshold values can be set on the basis of the number of cache updates.

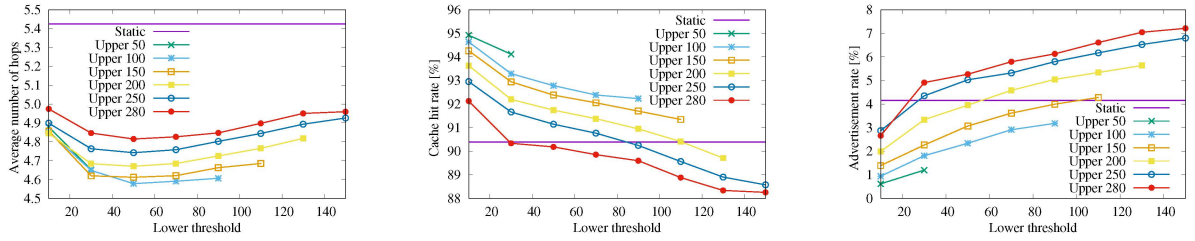
On the basis of the aforementioned discussion, we now show the simulation results and discuss the effectiveness of the proposed scheme as compared with the conventional schemes. Here, the lower/upper threshold values were set to 50/280, the reclustering interval was set to 3 seconds, and the initial cluster size of Hash-routing + cluster (HRC), conventional (Static), and proposed (Dynamic) schemes was set to 4, which was the appropriate value for an α of 1.0 at the start of the simulation. Figure 6 shows the average number of hops, cache hit rate, and cluster size as a function of time. From Figs. 6(a) and (b), the LCE scheme shows the worst performance among the other schemes because it causes duplicate caches on nearby CRs. The Hash-routing (HR) scheme improves the performance, especially cache efficiency, compared with the LCE scheme due to no duplicate cache occurrences, but the average number of hops, i.e., delivery latency is not good because the caches are distributed widely. The Hash-routing + cluster (HRC) scheme improves the performance compared with the HR scheme due to controlling the cache distribution range at the cost of a little cache efficiency. The conventional (Static) scheme using advertisement-based routing improves the performance compared with the HRC scheme due to the avoidance of detour routing caused by the false-positive problem with the HR scheme as well as the effect of retrieving nearby caches regardless of cluster boundaries. The proposed (Dynamic) scheme further improves the delivery latency while maintaining the cache hit rate compared with the conventional (Static) scheme in almost all ranges of time because it adjusts the cluster size to an adequate value.

Next, let us take a look at adjusting the cluster size of the Dynamic scheme focusing on three periods where the content popularity changes. First, in the period of 30–90 seconds, a wider range of content becomes to be requested, so that the cluster size is adjusted to a larger value (it is 6, which is an adequate value when α = 0.6 (Fig. 5(a))) due to the high frequency of cache updates as shown in Fig. 6(c). It improves the cache hit rate as well as delivery latency, although it takes time to distribute new caches in the cluster. Second, in the period of 90–150 seconds, the requested content becomes to be concentrated, so that the cluster size is adjusted to a smaller value (it is 2, which is an adequate value when α = 1.8) due to the low frequency of cache updates. It improves the delivery latency, although it takes time to discard unnecessary caches from the cluster, and comes at the cost of a slight decrease in cache hit rate. Finally, in the period of 150–270 seconds, similar to 30–90 seconds the requested content becomes to be a wider range gradually, so that the cluster size is adjusted to larger values (they are 3 and 4, which are adequate values when α = 1.4 and 1.0, respectively). It improves delivery latency while maintaining a high cache hit rate. This adjustment of cluster size is performed by searching for the cluster size that keeps the number of cache updates in the range of 50 to 280. Therefore, the proposed scheme can adapt effectively to the environment where content popularity changes.



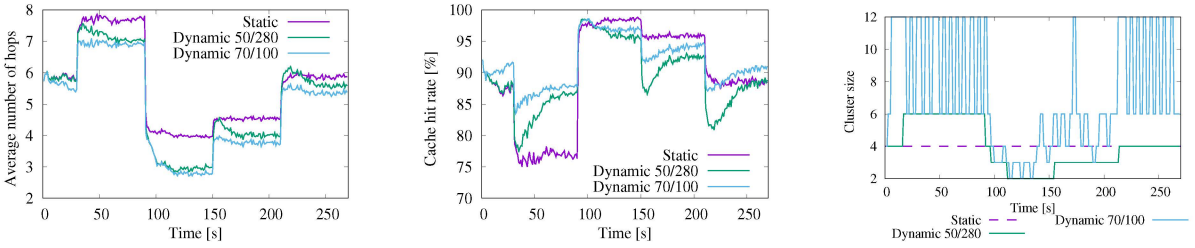
(a) Average number of hops (b) Cache hit rate (c) Cluster size

Fig. 6: Effectiveness of our scheme



(a) Average number of hops (b) Cache hit rate (c) Advertisement rate

Fig. 7: Effect of thresholds



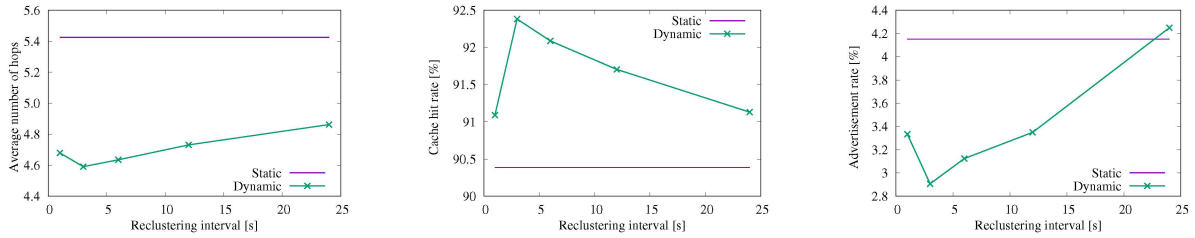
(a) Average number of hops (b) Cache hit rate (c) Cluster size

Fig. 8: Estimated and adequate thresholds

5.2 Effect of Thresholds

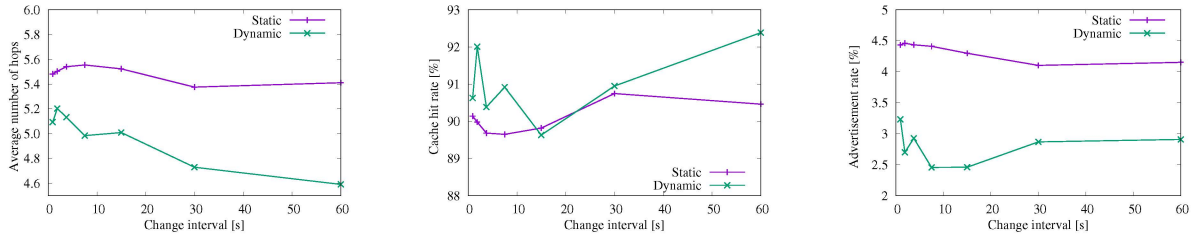
Next, we investigate the effect of the thresholds. Figures 7(a), (b), and (c) show the average number of hops, cache hit rate, and advertisement rate, respectively, when the lower/upper thresholds vary. Here, the reclustering interval was set to 3 seconds. Figures 7(a) and (b) indicate that the upper and lower threshold values should be set to an appropriate range (neither too large nor too small) to reduce delivery latency and maintain the high cache hit rate. When the upper threshold value is too large, it is difficult to migrate to a larger cluster size despite high frequent cache updates. As a result, it worsens cache efficiency as well as delivery latency. When the upper threshold value is too small, it is easy to migrate to a larger cluster size despite low frequent cache updates. As a result, it improves cache efficiency but increases delivery latency because the caches are widely distributed. The lower threshold observes a similar trend. Consequently, the adequate threshold values should be determined on the basis of the delivery latency and cache hit rate considering these trade-offs. The adequate lower/upper thresholds are 70/100 in this simulation environment, which achieves the lowest number of hops (Fig. 7(a)) and the high cache efficiency (Fig. 7(b)). Furthermore, the adequate cluster size does not cause frequent cache updates and reduces the flooding of advertisement packets for dynamic FIB updates, so the proposed (Dynamic) scheme with adequate thresholds also improves the advertisement rate, i.e., communication overhead, to approximately 2% of the total amount of traffic (Fig. 7(c)).

Here, it is noted that the estimated threshold values and adequate ones are largely different. This indicates that it should aggressively migrate to various sizes of clusters with the setting of larger/smaller lower/upper threshold values to maintain cache hit rates in the environment where the content popularity changes significantly. Figure 8 shows the average number of hops, cache hit rate, and change of cluster



(a) Average number of hops (b) Cache hit rate (c) Advertisement rate

Fig. 9: Effect of reclustering intervals



(a) Average number of hops (b) Cache hit rate (c) Advertisement rate

Fig. 10: Effect of change intervals of Zipf α

size in the conventional (Static) scheme and proposed (Dynamic) scheme with the estimated (50/280) and adequate (70/100) threshold values. Figure 8(c) clearly shows that the proposed scheme with adequate thresholds can more frequently migrate closer to the appropriate cluster size than that with estimated thresholds. Moreover, Fig. 8(a) and (b) show that such migration quickly improves the delivery latency and cache hit rate when the content popularity changes. Consequently, although the proposed scheme with estimated thresholds achieves good performance, it can be further improved by setting adequate thresholds on the basis of the aforementioned trade-offs as well as detecting sensitive changes in content popularity to quickly adjust the cluster size with appropriate cache distribution. However, the adequate threshold values may need to be adjusted dynamically in accordance with network conditions (the topology, frequency of requests, etc.), which will be tackled in future work.

5.3 Effect of Reclustering Intervals

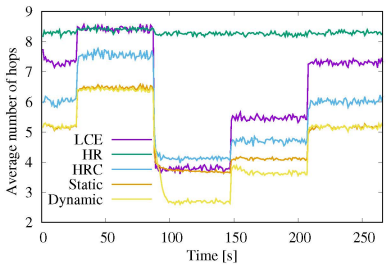
We investigate the effect of reclustering intervals. Figure 9 shows the average number of hops, cache hit rate, and advertisement rate when the reclustering interval varies. Here, the lower/upper thresholds were set to 70/100 (adequate values in this environment). Figure 9(a) shows that shorter reclustering intervals improve delivery latency except for too-short ones. This is because the shorter intervals can quickly migrate to the adequate cluster size and improve cache hit rates as shown in Fig. 9(b). However, too-short intervals inhibit migration to the adequate cluster size due to heavy cache updates immediately after reclustering. In addition, Fig. 9(c) shows that shorter reclustering intervals improve the overhead. This is because unnecessary cache updates are reduced by quickly migrating to the adequate cluster size. Consequently, the reclustering interval should be set to an adequately short value, which is 3 seconds in this environment.

5.4 Effect of Change Intervals of Zipf

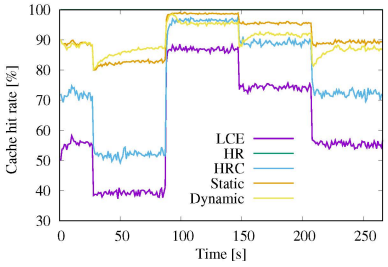
We investigate the effect of change intervals of Zipf α to show the environmental tolerance. For example, when the change intervals are set to 20 seconds, 30 seconds after the simulation starts with Zipf α of 1.0 and a cluster size of 4, Zipf α sequentially changes to 0.6, 1.4, 1.8, and 1.0 every 20 seconds, and these changes are repeated for 240 seconds (until the end of simulation). Figure 10 shows the average number of hops, cache hit rate, and advertisement rate when the change intervals of Zipf α vary. Here, the thresholds of lower/upper were set to 70/100, and the reclustering interval was set to 3 seconds (adequate values in this environment). From Fig. 10, the proposed (Dynamic) scheme always improves the delivery latency, cache hit rate, and overhead compared with the conventional (Static) scheme in a wide range of change intervals. This is because the proposed scheme can adapt cluster sizes smoothly to environments where the content popularity changes frequently.

5.5 Effect of Network Topology

Finally, we evaluate the proposed and conventional schemes comparatively in a practical network topology. We used the Interoute topology of 110 nodes from the Internet Topology Zoo [24] on the basis of comparative paper [8]. Since the



(a) Average number of hops



(b) Cache hit rate

Fig. 11: Effect of network topology

dataset shows the relationship of pop-level routers, we defined each node as CR and placed producers and consumers on each CR. Content was randomly placed on each producer. Each consumer sent interest packets requesting content toward the producer at normal distribution intervals with an average value of 1.0 seconds. The Dynamic scheme used the clustering algorithm described in Sec. 3.1 and its initial cluster size was set to approximately 4 (16 CRs in each cluster). The HRC scheme formed 6 clusters by the k-split algorithm. These settings were the appropriate value for an α of 1.0 at the start of the simulation. The Dynamic scheme can migrate the cluster size, which consists of 4, 9, 16, 36, or 110 CRs in each cluster, during the simulation. The reclustering interval was set to 3 seconds, the flooding limit was set to 6, and the lower/upper threshold values were set to 70/130, which were the appropriate values in this simulation. Other simulation parameters shall conform to Table 1.

Figure 11 shows the average number of hops and cache hit rate as a function of time. It indicates that the trend is almost the same as the results for the grid topology shown in Section 5.1, and the Dynamic scheme always maintains the high cache hit rates and reduces the average number of hops. In addition, regarding communication overheads, the Dynamic scheme achieves smaller advertisement rates of 4.37% than the Static scheme of 6.07%. Therefore, the proposed scheme is effective even in practical network topologies.

6. Conclusion

We proposed a dynamic clustering scheme to adjust the cache distribution range in accordance with the change in content popularity. Our scheme adjusts the cluster size effectively using a simple threshold-based algorithm based on the number of cache updates in the cluster. Simulation evaluations have indicated that the proposed scheme can reduce the delivery latency while consistently maintaining a high cache hit rate in a large domain environment where content popularity changes. In future work, we will investigate more flexible clustering schemes considering content attributes in practical topologies.

Acknowledgments

This work was supported in part by JSPS KAKENHI Grant-in-Aid for Scientific Research (C) Number 21K11872.

References

[1] M. Yoshida, Y. Ito, Y. Sato, and H. Koga, “Popularity-aware dynamic-clustering scheme for distributed caching in ICN,” Proc. ACM ICN2022, pp. 192–193, Sept. 2022. DOI:10.1145/3517212.3559482

[2] M. Yoshida, Y. Ito, Y. Sato, and H. Koga, “Performance evaluation of popularity-aware dynamic-clustering scheme for distributed caching in ICN,” Proc. APSIPA ASC2022, pp. 185–190, Nov. 2022.

[3] V. Jacobson, D.K. Smetters, J.D. Thornton, M. Plass, N. Briggs, and R. Braynard, “Networking named content,” Communications of the ACM, vol. 55, no. 1, pp. 117–124, Jan. 2012. DOI:10.1145/2063176.2063204

[4] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, “Named data networking,” ACM SIGCOMM Computer Communication Review, vol. 44, no. 3, pp. 66–73, Jul. 2014. DOI:10.1145/2656877.2656887

[5] S. Arshad, M.A. Azam, M.H. Rehmani, and J. Loo, “Recent advances in information-centric networking-based internet of things (ICN-IoT),” IEEE Internet of Things Journal, Vol. 6, no. 2, pp. 2128–2158, Apr. 2019. DOI:10.1109/JIOT.2018.2873343

[6] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, and S.Al-Ahmadi, “Named data networking: A promising architecture for the internet of things (IoT),” International Journal on Semantic Web and Information Systems, Vol. 14, no. 2, pp. 86–112, Apr. 2018. DOI:10.4018/IJSWIS.2018040105

[7] M. Yoshida, Y. Ito, Y. Sato, and H. Koga, “A cluster-based cache distribution scheme in content-centric-networking,” Proc. ACM ICN2018, pp. 196–197, Sept. 2018. DOI:10.1145/3267955.3269012

[8] V. Sourlas, I. Psaras, L. Saino, and G. Pavlou, “Efficient hash-routing and domain clustering techniques for information-centric networks,” Elsevier Computer Networks, vol. 103, pp. 67–83, July 2016. DOI:10.1016/j.comnet.2016.04.001

[9] N. Laoutaris, H. Che, and I. Stavrakakis, “The LCD interconnection of LRU caches and its analysis,” Elsevier Performance Evaluation, vol. 63, no. 7, pp. 609–634, July 2006. DOI:10.1016/j.peva.2005.05.003

[10] N. Laoutaris, S. Syntila, and I. Stavrakakis, “Meta algorithms for hierarchical web caches,” Proc. IEEE IPCCC2004, pp. 445–452, Apr. 2004. DOI:10.1109/PCCC.2004.1395054

[11] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, “Wave: Popularity-based and collaborative in-network caching for content-oriented networks,” Proc. IEEE INFOCOM2012 Workshops, pp. 316–321, May 2012. DOI:10.1109/INFCOMW.2012.6193512

[12] T. Mick, R. Tourani, and S. Misra, “MuNCC: Multi-hop neighborhood collaborative caching in information centric networks,” Proc. ACM ICN2016, pp. 93–101, Sept. 2016. DOI:10.1145/2984356.2984375

[13] L. Saino, I. Psaras, and G. Pavlou, “Hash-routing schemes for information centric networking,” Proc. ACM ICN2013, pp. 27–32, Aug. 2013. DOI:10.1145/2491224.2491232

[14] H.M. Ju and L. Hyesook, "Cache sharing using Bloom filters in named data networking," Journal of Network and Computer Applications, vol. 90, pp. 74–82, July 2017. DOI:10.1016/j.jnca.2017.04.011

[15] C. Li and K. Okamura, "Cluster-based in-networking caching for content-centric networking," International Journal of Computer Science and Network Security, vol. 14, no. 11, pp. 1–9, 2014. http://paper.ijcsns.org/07_book/201411/20141101.pdf

[16] B. Alahmri, S. Al-Ahmadi and A. Belghith, "Efficient pooling and collaborative cache management for NDN/IoT networks," IEEE Access, vol. 9, pp. 43228-43240, Mar. 2021. DOI:10.1109/ACCESS.2021.3066133

[17] H. Yan, D. Gao, W. Su, C.H. Foh, H. Zhang and A.V. Vasilakos, "Caching strategy based on hierarchical cluster for named data networking," IEEE Access, vol. 5, pp. 8433-8443, Mar. 2017. DOI:10.1109/ACCESS.2017.2694045

[18] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini, "Temporal locality in today's content caching: why it matters and how to model it," SIGCOMM Computer Communication Review, vol. 43, no. 5, pp. 5–12, Oct. 2013. DOI:10.1145/2541468.2541470

[19] W. Wong, L. Wang, and J. Kangasharju, "Neighborhood search and admission control in cooperative caching networks," Proc. IEEE GLOBECOM2012, pp. 2852–2858, Dec. 2012. DOI:10.1109/GLOCOM.2012.6503549

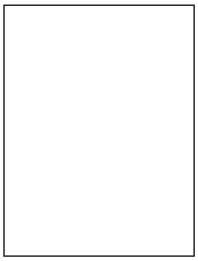
[20] S. Nayak, R. Patgiri, A. Borah, "A survey on the roles of Bloom Filter in implementation of the Named Data Networking," Elsevier Computer Networks, vol. 196, art. no. 108232, Sept. 2021. DOI:10.1016/j.comnet.2021.108232

[21] G.F. Riley, and T.R. Henderson, "The ns-3 Network Simulator," in Modeling and Tools for Network Simulation, ed. K. Wehrle, M. Güneş, J. Gross, pp. 15–34, Springer, Berlin, Heidelberg, 2010. DOI:10.1007/978-3-642-12331-3_2

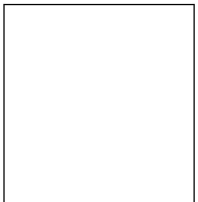
[22] A. Ioannou, and S. Weber, "A survey of caching policies and forwarding mechanisms in information-centric networking," IEEE Communications Surveys & Tutorials, vol. 18, no. 4, pp. 2847–2886, May 2016. DOI:10.1109/COMST.2016.2565541

[23] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," IEEE/ACM Transactions on Networking, vol. 16, no. 6, pp. 1447–1460, Mar. 2008. DOI:10.1109/TNET.2008.918081

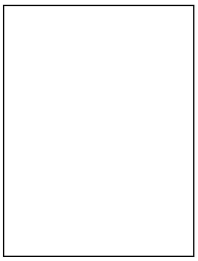
[24] S. Knight, H. Nguyen, N. Falkner, R. Bowden, M. Roughan, "The internet topology zoo," IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pp. 1765–1775, Oct. 2011. DOI:10.1109/JSAC.2011.111002



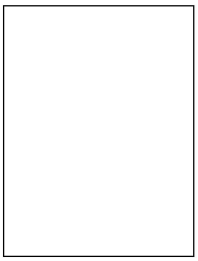
Mikiya Yoshida received the B.E. degree in Information Science and Electrical Engineering from the Kyushu Sangyo University, Japan, in 2017, and his M.E. degree in Information Engineering from the University of Kitakyushu, Japan, in 2019. He is now a doctoral student at the University of Kitakyushu, Japan. His research interests include network architecture and information-centric networking.



Yusuke Ito received the B.E., M.E., and D.E. degrees in Information and Media Engineering from the University of Kitakyushu, Japan in 2014, 2016, and 2019, respectively. From 2019 to 2022, he was an Assistant Professor at the Tokyo University of Science. Currently, he is a Lecturer in the Department of Information Systems Engineering, Faculty of Environmental Engineering, The University of Kitakyushu, Japan. His research interests include network architecture and edge cloud computing.



Yurino Sato received the B.E., M.E., and D.E. degrees in Information and Media Engineering from the University of Kitakyushu, Japan in 2012, 2014, and 2019, respectively. She has been an assistant professor since April 2018 in the Department of Control Engineering, National Institute of Technology (KOSEN), Sasebo College, Japan. Her research interests include network architecture, transport protocol, and forward error correction.



Hiroyuki Koga received the B.E., M.E., and D.E. degrees in Computer Science and Electronics from the Kyushu Institute of Technology, Japan in 1998, 2000, and 2003, respectively. From 2003 to 2004, he was a postdoctoral researcher in the Graduate School of Information Science, Nara Institute of Science and Technology. From 2004 to 2006, he was a researcher in the Kitakyushu JGN2 Research Center, National Institute of Information and Communications Technology. From 2006 to 2009, he was an assistant professor in the Department of Information and Media Engineering, Faculty of Environmental Engineering, The University of Kitakyushu, and has been an associate professor in the same department since April 2009. His research interests include performance evaluation of computer networks, mobile networks, and communication protocols.