Regular Paper

Avoiding Excessive Rate Reduction in Rate Based Congestion Control for Named Data Networking

Takahiko Kato[1,a) Masaki Bandai[2]

Received: April 12, 2017, Accepted: October 3, 2017

Abstract: In the paper, we propose a new congestion control method based on hop-by-hop rate control in named data networking (NDN). The proposed method suppresses the excessive reduction of the Interest sending rate due to continuous negative acknowledgement (NACK) packets caused by the propagation delay. The proposed method limits the rate reduction to once per congestion. The point of the proposed method is to keep the Interest sending rate for NACK packets unchanged after the rate reduction for congestion until the rate reduction becomes effective. Performance evaluation is conducted through computer simulations in order to demonstrate that the proposed method improves the throughput performance as compared to the conventional method.

Keywords: named data networking, congestion control, hop-by-hop rate control

1. Introduction

In recent years, content delivery traffic, such as video streaming, has increased significantly, and the user requirement for Internet communications has become more content-oriented rather than location-oriented. However, the current Internet architecture focuses on content location. In order to address this gap, the information centric network (ICN) [1], [2], [3] has attracted attention as a new Internet architecture that is suitable for retrieving and delivering content. Named data networking (NDN) [4] is one of the widely studied ICN architectures. In NDN, a content requester (consumer) uses the name of the content rather than the IP address of the node publishing the content (producer). NDN uses two types of packets: Interest and Data. A consumer that requests a specific content sets the name of the content in an Interest packet and sends it toward the content producer. The producer returns the Data packet to the consumer in response to the received Interest packet. One Interest packet requests one Data packet and a Data packet is transferred through the reverse path of the corresponding Interest packet. While transferring a Data packet, intermediate NDN routers (routers for short) cache the Data packet for distribution in the future.

If an Interest packet or a Data packet arriving at a router exceeds its forwarding capability, it will experience a delay or a loss at the router. In order to prevent this problem, the congestion control mechanism for NDN architecture is widely investigated. In the traditional Internet, the congestion control is often discussed together with the flow control. However, since NDN adopts the receiver driven principle and one-Interest-one-Data transport operations, the Data packet traffic can be controlled by adjusting the sending rate of Interest packets at the receiver. Therefore, in general, the flow control in NDN is easy [5]. On the other hand, a consumer cannot estimate the resource usage in intermediate routers and links. Therefore, the congestion control is an important topic in NDN. NDN congestion control approaches can be categorized as end-based window or hop-by-hop rate control approaches [5]. End-based window control is the traditional TCP-like mechanism. A consumer, as an end node, determines its window size for sending Interest packets and estimates congestion based on the round trip time (RTT), which is the duration between an Interest packet and the corresponding Data packet [6], [7], [8]. Since intermediate routers do not control the packet rate, this approach is easy to implement. However, in NDN, the RTT changes significantly due to the cache mechanism in intermediate routers. Thus, it is difficult to detect congestion correctly. Therefore, the end-based window control is not suitable for NDN [9]. On the other hand, in hop-by-hop rate control, a consumer and routers control the Interest sending rate based on the local information, such as the link bandwidth [10], [11], [12], [13]. Since hop-by-hop rate control does not depend greatly on the RTT, it is more suitable for NDN than end-based window control [9].

The hop-by-hop approach still has some problems when congestion occurs. A consumer or router detects congestion by means of a timeout while waiting for a Data packet, or by receiving a negative acknowledgment (NACK) packet. Timeout-based detection cannot correctly estimate congestion because of the fluctuation of the RTT. Moreover, timeout-based detection cannot quickly respond to congestion. On the other hand, in NACK-based detection, a router sends a NACK packet at the moment it detects congestion. Therefore, it is considered that NACK-based detection can resolve the problems of timeout-based detection. However, there is another problem. Since a propagation delay

1 Graduate School of Science and Technology, Sophia University, Chiyoda, Tokyo 102–0071, Japan
2 Dept. of Information and Communication Science, Sophia University, Chiyoda, Tokyo 102–0071, Japan
a) tk-kt0518@eagle.sophia.ac.jp

exists between the router that detects the congestion and its neighbor router, the neighbor router continues to send Interest packets at the rate that caused the congestion until it receives the NACK packet. These Interest packets trigger further NACK packets. As a result, the neighbor router reduces the Interest sending rate multiple times, which reduces the rate excessively.

In this paper, we propose a new hop-by-hop rate control method that resolves the excessive rate reduction problem of the NACK-based approach. The proposed method limits the rate reduction to once per congestion. The goal of the proposed method is to keep the Interest sending rate for NACK packets unchanged after the rate reduction for congestion until the arrival of the first Data or NACK packet corresponding to the first Interest packet transferred at a reduced rate. Through the performance evaluation by computer simulations, we show that the proposed method improves the throughput performance compared to the conventional method. The basic concept of the proposed method was presented in Ref. [14], and the method is extended in this paper.

The remainder of the paper is organized as follows. We explain some existing research on NDN congestion control and the excessive rate reduction problem in Section 2. The proposed method is described in Section 3. Section 4 discusses the results of performance evaluation using the ndnSIM network simulator. Section 5 concludes the paper.

2. Existing Research and Associated Problem

2.1 Existing Research on NDN Congestion Control

In the end-based window control approach, a consumer sends Interest packets with the limitation of window size. The Interest control protocol (ICP) [7], the information centric transport protocol (ICTP) [8], and the content centric TCP (CCTCP) [9] are examples of the end-based window control approach. As described above, these methods depend on RTT, which changes significantly. Thus, it is difficult to determine the optimal value of the window size. Moreover, the timeout value for detecting congestion is not difficult to set.

Several studies have examined the hop-by-hop rate control approach. The hop-by-hop Interest shaping (HoBHIS) [10] decides how routers determine the Interest sending rate based on the link bandwidth, the transmission buffer of Data packets, and RTT. In the hop-by-hop and receiver-driven Interest control protocol (HR-ICP) [11], routers use per-content-flow counters and buffers in order to control the balance of Interest and Data packets. Stateful Forwarding [12] controls the Interest sending rate based on the link bandwidth and the size of a Data packet. A method for determining the Interest sending rate by taking into account the overhead of Interest packets has been proposed [13].

As for the congestion detection, HoBHIS and HR-ICP use a timeout-based method. If a Data packet is not received during the timeout period because the router or consumer sends an Interest packet, the router determines that congestion has occurred and reduces the Interest sending rate. However, since the timeout period is determined based on RTT, a problem similar to that encountered for the end-based window control approach exists.

On the other hand, NACK-based detection is used in Stateful Forwarding as well as a previous study [13]. A NACK packet is

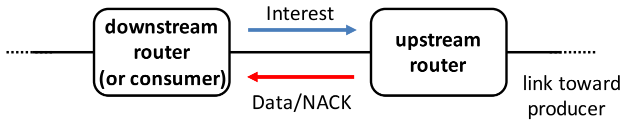


Fig. 1 Congestion notification in NACK based detection.

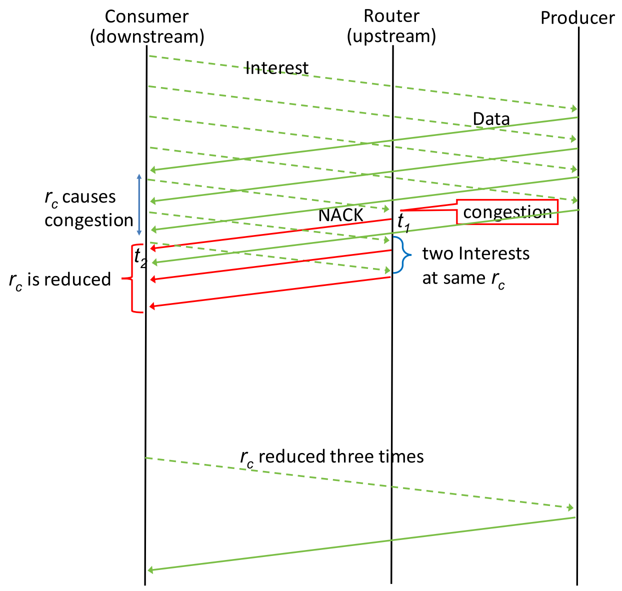


Fig. 2 Example sequence of excessive rate reduction problem.

sent to notify the neighbor router (or the consumer) in the downstream side of congestion in the link toward the producer. **Figure 1** shows the relationship between the downstream router (or consumer) and the upstream router. If the upstream router receives Interest packets at a fast rate that causes congestion, the router returns a NACK packet to the downstream router. In this way, the downstream router can quickly detect the occurrence of congestion and then reduce the Interest sending rate.

2.2 Excessive Rate Reduction Problem

The rate control mechanism with NACK-based congestion detection has a problem that the Interest sending rate is reduced excessively due to the propagation delay of NACK packets. In Fig. 1, the propagation delay is the time gap between the time when a NACK packet is sent from the upstream router and the time when the packet arrives at the downstream router. During the propagation delay, the downstream router maintains the Interest sending rate that caused the congestion. As a response to these Interest packets, the upstream router still sends NACK packets consecutively because the congestion continues. As a result, the downstream router receives more than one NACK packet for the same Interest sending rate and reduces the rate in response to these NACK packets. The downstream router does not check whether the once reduced Interest sending rate resolves the congestion or not. Thus, reducing the rate multiple times may result in the rate being too low for the current network condition.

Figure 2 shows an example of this problem. We assume that the Interest sending rate of the consumer, denoted by r_c in the figure, becomes higher than that of the router at time t_1. The router detects congestion at time t_1 and returns a NACK packet to the consumer. The NACK packet reaches the consumer at time t_2.

Before reducing r_c at time t_2, the consumer sends two more Interest packets at the same value of r_c which causes congestion at the Router. The router returns a total of three NACK packets to the consumer. The consumer reduces r_c three times during one congestion, which is considered to be an excessive reduction.

3. Proposed Method

In order to resolve the excessive rate reduction problem, we adopt an approach in which once a downstream router (or a consumer) receives a NACK packet and reduces the Interest sending rate, the router does not change the rate until confirming that the upstream router (or the producer) has responded to this rate reduction. More specifically, we use the following scheme. After a downstream router receives a NACK packet and reduces the Interest sending rate, the router waits for the Data or NACK packet corresponding to the Interest packet that the router sent at the reduced rate. During the period from the rate reduction to the receipt of this Data or NACK packet, the downstream router does not change the Interest sending rate when it receives a packet.

The proposed method introduces three states for each interface and content name: *normal*, *congestion* and *check*. The description and transition of each state is as follows:

(1) *Normal*: The *normal* state indicates that the network works normally and that no congestion occurs. In this state, a router controls the Interest sending rate according to the original control method, such as the additive increase and multiplicative decrease (AIMD) mechanism. When a router receives a Data packet, the router increases the rate. When a router receives a NACK packet, the router decreases the rate and transits to the *congestion* state.

(2) *Congestion*: The *congestion* state means that congestion occurs. In this state, a router does not change the Interest sending rate when the router receives a Data or NACK packet. When the router sends an Interest packet, the router records the sending time of the Interest packet, which we refer to as T_{send}, and then transits to the *check* state.

(3) *Check*: In the *check* state, a router checks whether the congestion ends. When the router receives a Data or NACK packet, the router checks its pending interest table (PIT) [4] to obtain the sending time of the Interest packet corresponding to the received Data or NACK packet. We refer to the time as T_{int}. The router compares T_{int} and T_{send}.
- If T_{int} is smaller than T_{send}, the router does not change the Interest sending rate.
- If T_{int} is greater than or equal to T_{send}, the router checks whether the received packet is a Data or a NACK packet.
  - If the packet is a Data packet, the router increases the rate and transits to the *normal* state because the router realizes that the congestion ends.
  - If the packet is a NACK packet, the router decreases the rate and transits to the *congestion* state because the router realizes that the congestion continues.

Figure 3 illustrates an example of the behavior of the proposed method. This example is under the same situation as Fig. 2. Initially, the consumer state is *normal*. The router returns a NACK packet to Consumer at time t_1. When the consumer receives the NACK packet at time t_2, the router reduces the Interest sending rate (r_c) and transits to *congestion*. At times t_3 and t_4, the consumer receives one NACK packet and one Data packet, respectively. The consumer does not change r_c in both cases because the state is *congestion*. Then, at time t_5, the consumer sends an Interest packet at the reduced r_c. At that moment, the router records the sending time T_{send} and transits to *check*. At time t_6, the consumer receives a NACK packet. The router does not reduce r_c for this NACK packet. Finally, at time t_7, the consumer receives a Data packet for which the corresponding Interest sending time T_{int} is equal to T_{send}. At that moment, the consumer realizes that the congestion ends and transits to *normal*. In this way, the proposed method limits the rate reduction to once per congestion.

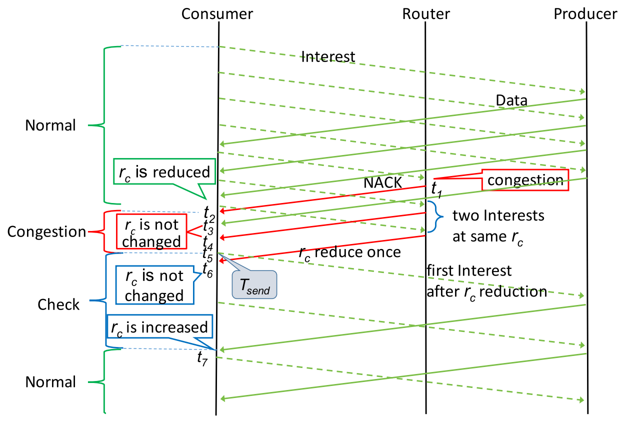


Fig. 3 Behavior of proposed method in congestion.

4. Performance Evaluation

4.1 Simulation Environment

4.1.1 Comparative System

In this paper, we select Stateful Forwarding as a comparative method. We implement the proposed method based on the Stateful Forwarding. In this subsection, we describe an overview of Stateful Forwarding.

In the rate control of Stateful Forwarding, a consumer or router manages the Interest sending rate limit L [packet/s] and the Interest sending rate r. L is defined as follows for each interface:

$$L = \frac{C}{S}, \tag{1}$$

where C [bps] is the link bandwidth of the interface, and S [bit/packet] is the estimated size of Data packets that have been received over the interface. This ratio is the maximum Data receiving rate from the upstream router (or the producer). Therefore, L is used as the maximum Interest rate going upstream. On the other hand, a consumer or router also manages the Interest sending rate r [packet/s] for each interface and content name, where r changes according to the congestion situation.

When the upstream router, as shown in Fig. 1, receives an Interest packet, the router checks the corresponding L and r. If the receiving rate of Interest packets is higher than one or both of L and r, the upstream router decides that congestion occurs and returns a NACK packet instead of a Data packet to the downstream router. Then, the downstream router decreases the value of r upon receiving the NACK packet. Therefore, Stateful For-

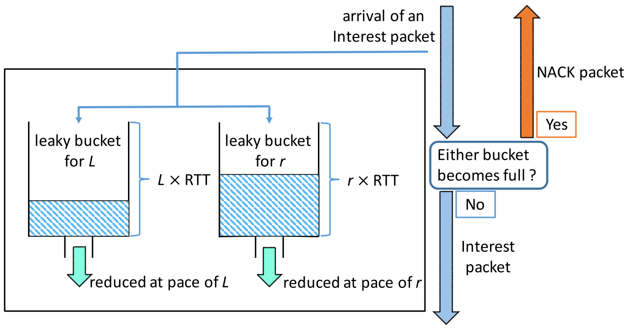


Fig. 4 Behavior of rate control using leaky bucket mechanism.

warding prevents Data packet loss by adjusting the Interest sending rate before buffer overflow occurs for Data packets. When the downstream router receives a Data packet, the router increases the value of r. One way to control r is the AIMD mechanism. In this paper, when a consumer receives a Data packet, r is incremented by 1 [packet/s] under the limit of L. When a consumer receives a NACK packet, r is halved.

Stateful Forwarding is implemented on the network simulator ndnSIM ver1.0 [14]. In order to implement the rate control, ndnSIM uses the leaky bucket mechanism. **Figure 4** shows its behavior. A router has two types of buckets: a bucket for L, corresponding to each interface, and a bucket for r, corresponding to each interface and content name. The depth of the bucket for L is the product of the RTT between the router and a producer and L, and the depth of the bucket for r is the product of the RTT between the router and a producer and r. When a router receives an Interest packet, it adds one packet to the corresponding leaky buckets for L and r. The router reduces packets from the leaky buckets at the pace of L or r. If either bucket for L or r becomes full when the router receives an Interest packet, the router returns a NACK packet to the downstream router. If neither the bucket for L nor r is full, then the router sends the Interest packet toward a producer. We modify the part of ndnSIM ver. 1.0 that supports r to work correctly.

4.1.2 Evaluation Conditions

We compare the performance of the proposed method and Stateful Forwarding. The evaluation conditions are as follows:

- We use two topologies in the simulation: linear topology and dumbbell topology. In the linear topology, an intermediate link has a smaller bandwidth than the other links, and this situation introduces a static congestion condition. Through this topology, we can evaluate the behavior of one content delivery in detail. In the dumbbell topology, two, four, and eight consumers deliver different contents from dedicated producers. In order to evaluate the performance when the traffic condition changes dynamically, some consumers send Intrest packets in the different duration from the others. The shared link is a bottleneck and generates congestion. Using this topology, we can evaluate congestion invoked by multiple content deliveries.
- The Data packet size is 1,250 bytes, the Interest packet size is 37 bytes and the transmission buffer size for Data packets is 200 packets.
- The depth of the leaky bucket is set to the default setting as

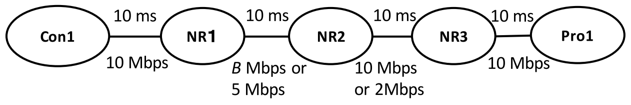


Fig. 5 Linear topology.

described above.

- The simulation time is 10 seconds.
- We do not consider in-network caching by intermediate routers, because we focus on the pure performance of the proposed method.

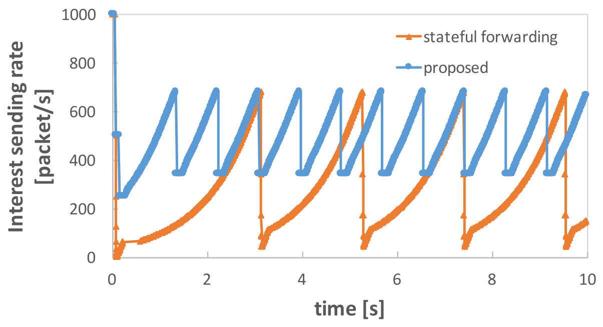
In order to analyze the detailed behavior of each method, we evaluate the Interest sending rate r_c and the Data throughput measured at the consumer. The Data throughput is defined as the number of Data packets that a consumer or a router receives in a second, which is measured every 0.1 seconds. We also evaluate the bottleneck link utilization, which is defined as the average Data throughput divided by the bottleneck link bandwidth.

4.2 Performance Evaluation Results

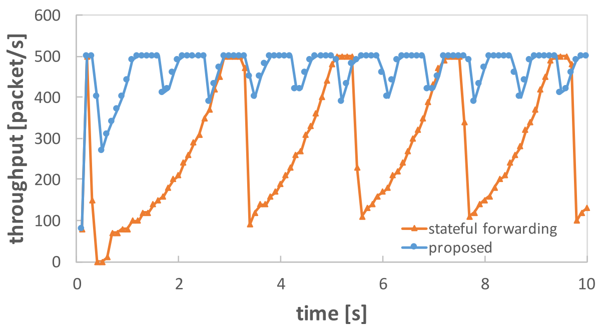
4.2.1 Linear Topology

As shown in **Fig. 5**, the linear topology is composed of one consumer (Con1), one producer (Pro1), and three routers (NR1, NR2 and NR3). Con1 downloads content from Pro1. Using this topology, we evaluate two cases. One is the case where there is one bottleneck link. The link between NR1 and NR2 has a bandwidth of B Mbps, which ranges from 1 Mbps to 9 Mbps. The other is the case where there are two bottleneck links. The link between NR1 and NR2 has a bandwidth of 5 Mbps and that between NR2 and NR3 2 Mbps.

The result for one bottleneck link is as follows. First, we explain the detailed behavior of each method for the cases in which $B = 5$ Mbps and $B = 1$ Mbps. We then explain the bottleneck link utilization of each method when varying B from 1 Mbps to 9 Mbps. **Figure 6** shows the Interest sending rate r_c and the Data throughput at Con1 when $B = 5$ Mbps. **Table 1** shows the number of received NACK packets during congestion at Con1. From Eq. (1), the Interest sending rate limits at Con1 and NR1 are 1,000 packet/s and 500 packet/s, respectively. These are the initial values and the maximum values of r_c and r_{n1}. As shown in Fig. 6 (a), the Interest sending rate drops multiple times in both methods. These drops are caused by congestion. In Stateful Forwarding, when congestion occurs, r_c decreases multiple times. r_c is reduced to around 1.5 packet/s at the first congestion and around 45 packet/s at other congestions. This occurs because Con1 halves r_c as many times as the number of received NACK packets shown in Table 1 (a). Since r_{n1} is 500 packet/s, it can be said that r_c is reduced excessively. In the proposed method, Table 1 (b) shows that the number of received NACK packets is similar to Stateful Forwarding. However, Fig. 6 (a) shows that when congestion occurs, r_c is reduced only once (twice at the first congestion) for continuously received NACK packets. Except the first congestion, after reducing r_c once in each congestion, Con1 does not change r_c for approximately 0.15 seconds, which is the duration of the *congestion* and *check* states. Thus, in the proposed method, since Con1 does not change r_c until the reduced r_c value is confirmed to resolve the congestion, r_c is not reduced multiple

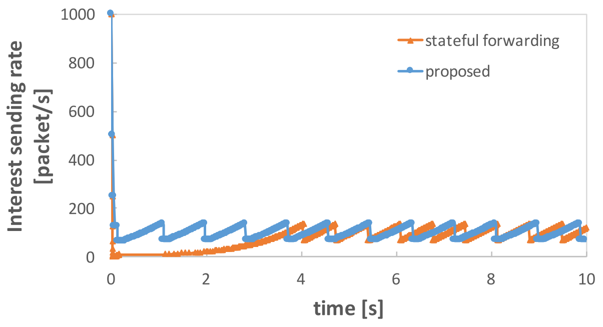


(a) Interest sending rate at Con1

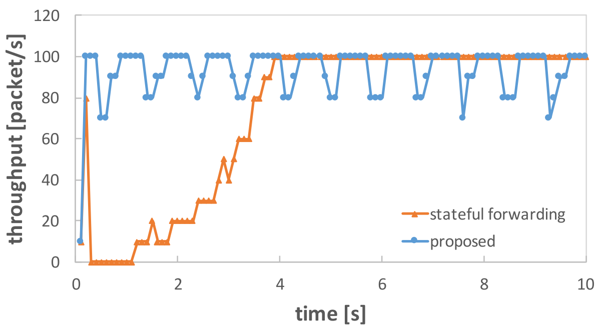


(b) Data throughput at Con1

Fig. 6 Interest sending rate and Data throughput when B = 5 Mbps in linear topology.



(a) Interest sending rate at Con1



(b) Data throughput at Con1

Fig. 7 Interest sending rate and Data throughput when B = 1 Mbps in linear topology.

Table 1 Number of received NACK packets during congestion when B = 5 Mbps in linear topology.

(a) Stateful Forwarding

Start time of congestion (seconds)	0.08	3.13	5.27	7.40	9.54
Number of NACK packets	11	4	4	4	4

(b) Proposed method

Start time of congestion (seconds)	0.08	1.35	2.23	3.09–9.15
Number of NACK packets	12	4	4	4

Table 2 Number of received NACK packets during congestion when B = 1 Mbps in linear topology.

(a) Stateful Forwarding

Start time of congestion (seconds)	0.03	4.05	4.74	5.42–9.50
Number of NACK packets	21	1	1	1

(b) Proposed method

Start time of congestion (seconds)	0.03	1.10	1.98	2.84–9.88
Number of NACK packets	35	1	1	1

times continuously. Since the value of r_c halved once is equal to or less than r_{n1} which is the rate of the bottleneck link, Con1 does not need to reduce r_c any further. In other words, one rate reduction is sufficient to resolve the congestions.

From Fig. 6 (b), the Data throughput of the proposed method is stable and the average Data throughput is 465 packet/s. Since the maximum Data rate in the bottleneck link is 500 packet/s, the bottleneck link utilization of the proposed method is 93 percent. On the other hand, the Data throughput of Stateful Forwarding varies between 0 and 500 packet/s depending on the value of r_c, which varies between 1 and 500 packet/s. The average Data throughput is 274 packet/s, and the bottleneck link utilization is only 55 percent. Based on these results, the proposed method is effective for improving the throughput when B = 5 Mbps.

Figure 7 shows the Interest sending rate r_c and the Data throughput of Con1 when B = 1 Mbps. **Table 2** shows the number of received NACK packets during congestion at Con1. As shown in Fig. 7 (a), in Stateful Forwarding, at the first congestion, r_c decreases multiple times and is reduced to 1 packet/s, which is the lowest rate. However, from the second congestion, r_c is reduced once for each congestion because the number of received NACK packets is 1 packet. This is caused by a low r_c value. On the other hand, the proposed method also works well in this case. In the congestion at 0.03 seconds, r_c is halved four times. After halving r_c once, Con1 transits to the *congestion* and *check* states and does not change r_c. The reduced value of r_c is 500 packet/s and is higher than r_{n1}, which is 100 packet/s. Thus, Con1 receives a NACK packet again for an Interest packet sent at 500 packet/s. Finally, r_c is halved four times and becomes 62.5 packet/s, which is lower than r_{n1}. For the other congestions, r_c is halved once. Thus, the proposed method can reduce the Interest sending rate appropriately. From Fig. 7 (b), the average Data throughput in the proposed method is 93 packet/s, and thus its bottleneck link utilization is 93 percent. The Data throughput in Stateful Forwarding varies between 0 and 100 packet/s, and the average Data throughput is 73 packet/s. This means that the bottleneck link utilization is 73 percent in the topology. The proposed method works well for the case in which B = 1 Mbps.

Figure 8 shows the bottleneck link utilization for various bottleneck link bandwidths between 1 Mbps and 9 Mbps. As shown

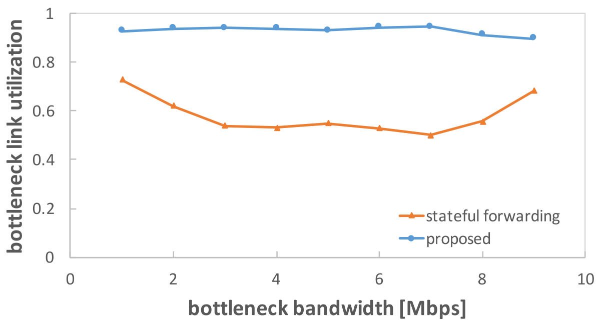
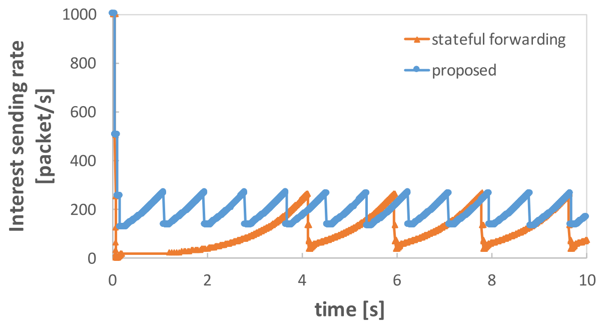
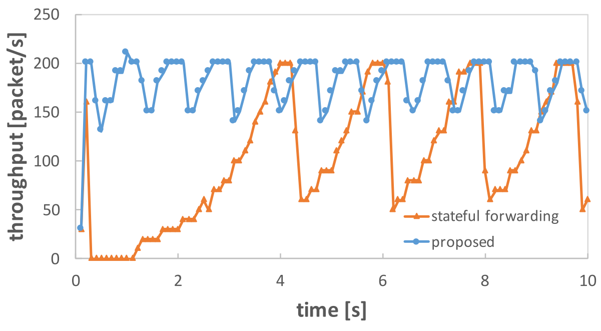


Fig. 8 Bottleneck link utilization in linear topology.



(a) Interest sending rate at Con1



(b) Data throughput at Con1

Fig. 9 Interest sending rate and Data throughput when there are two bottleneck links in linear topology.

in Fig. 8, Stateful Forwarding has a lower bottleneck link utilization of between 50 and 73 percent. This is because when congestion occurs, Con1 reduces the Interest sending rate multiple times regardless of the value of *B*. When *B* is 1 or 2 Mbps, the utilization of Stateful Forwarding increases because the number of received NACK packets per congestion decreases. When *B* is 8 or 9 Mbps, it also increases because the number of congestion decreases. On the other hand, the proposed method has a higher bottleneck link utilization of between 90 and 95 percent. Con1 appropriately reduces the Interest sending rate according to the value of *B*.

The result for two bottleneck links is as follows. **Figure 9** shows the Interest sending rate r_c and the Data throughput of Con1. Although there are two links with limited bandwidth, the link between NR2 and NR3 is smaller than that between NR1 and NR2. So, the result in Fig. 9 (a) has a similar tendency to the result when there is only one bottleneck link with 2 Mbps. In Stateful Forwarding, r_c decreases multiple times at congestions, and in the proposed method, r_c decreases only once at congestions

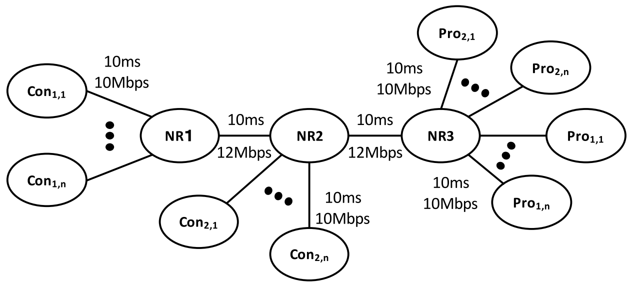


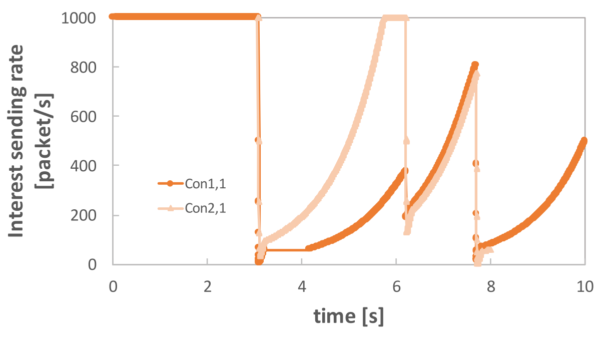
Fig. 10 Dumbbell topology.

except the first one. From Fig. 9 (b), the average Data throughput in the proposed method is 180 packet/s, and thus the bottleneck link utilization is 90 percent in this topology. The average Data throughput in Stateful Forwarding is 102 packet/s, and the bottleneck link utilization is 51 percent. Although the results of Data throughput are similar to those of one bottleneck link with 2 Mbps, the values of the bottleneck link utilization are smaller for both the proposed method and Stateful Forwarding. It is because the depth of the leaky bucket in NR2 is smaller than that in NR1, due to the smaller value of RTT.

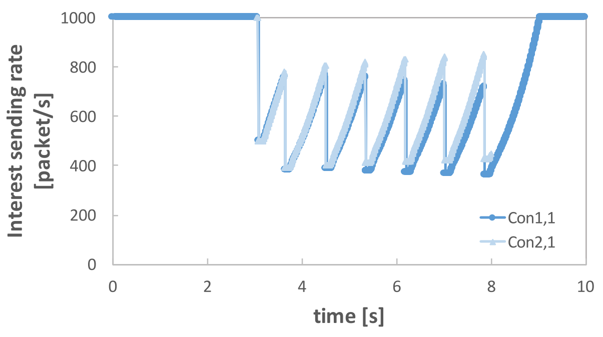
4.2.2 Dumbbell Topology

In this subsection, we show how the proposed method works when there are multiple flows. We use a dumbbell topology which has two NDN routers accommodating consumers. As shown in **Fig. 10**, the dumbbell topology is composed of $2n$ consumers ($Con_{1,1}$ through $Con_{1,n}$, and $Con_{2,1}$ through $Con_{2,n}$), $2n$ producers ($Pro_{1,1}$ through $Pro_{1,n}$, and $Pro_{2,1}$ through $Pro_{2,n}$), and three NDN routers (NR1, NR2 and NR3). $Con_{1,i}$ and $Con_{2,i}$ download an individual content from $Pro_{1,i}$, and $Pro_{2,i}$ respectively ($i = 1$ through n). $Con_{1,i}$ retrieves content from 0 to 10 seconds. $Con_{2,i}$ retrieves from time 3 to 8 seconds. The link between NR1 and NR2 and that between NR2 and NR3 become the bottleneck link. The link between NR2 and NR3 is more congested than that between NR1 and NR2. We explain the detailed behavior of each method when $n = 1$, 2 and 4.

Figures 11 and **12** show the results when $n = 1$. Figure 11 shows the Interest sending rate at $Con_{1,1}$ ($r_{c1,1}$) and that at $Con_{2,1}$ ($r_{c2,1}$). From Fig. 11 (a), in Stateful Forwarding, Con1 sends Interest packets at $r_{c1,1}$ = 1,000 packet/s before 3 seconds constantly. Since the Interest sending rate limit at Con1 ($L_{c1,1}$) is 1,000 packet/s, Con1 does not increase $r_{c1,1}$ even if it receives Data packets. During the period from 3 to 8 seconds, congestion occurs at 3.10, 6.21 and 7.70 seconds because the sum of $r_{c1,1}$ and $r_{c2,1}$ is larger than L_{n2} (= 1,200 packet/s). Both $r_{c1,1}$ and $r_{c2,1}$ decrease multiple times. Since $Con_{1,1}$ and $Con_{2,1}$ can send at the rate of 600 packet/s if the bottleneck link is shared equally, $r_{c1,1}$ and $r_{c2,1}$ are reduced excessively. There is a difference between $r_{c1,1}$ and $r_{c2,1}$. It is because the number of received NACK packets at $Con_{1,1}$ differs from that at $Con_{2,1}$. This NACK number difference comes from the difference in the delay between each consumer and a congested router (NR2), and the difference between $r_{c1,1}$ and $r_{c2,1}$ when congestion occurs. After time 8 seconds, r_{c1} increases up to 1,000 packet/s. In the proposed method, Fig. 11 (b) shows that when congestion occurs, both $r_{c1,1}$ and $r_{c2,1}$ are halved only once. This is because the

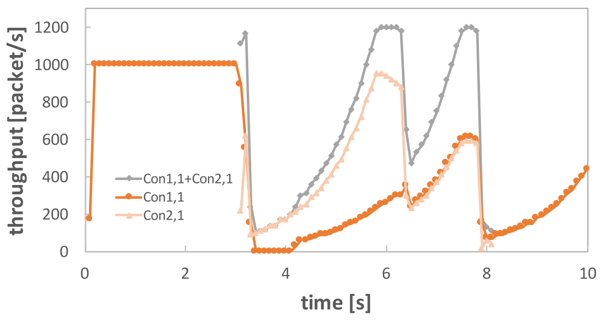


(a) Stateful Forwarding

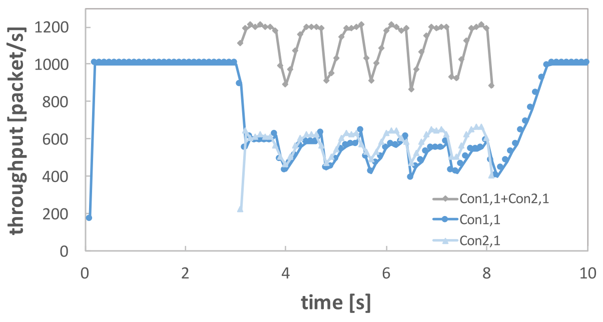


(b) Proposed method

Fig. 11 Interest sending rates in dumbbell topology when n = 1.



(a) Stateful Forwarding

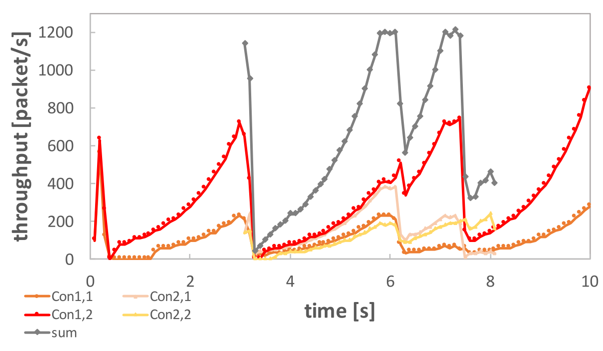


(b) Proposed method

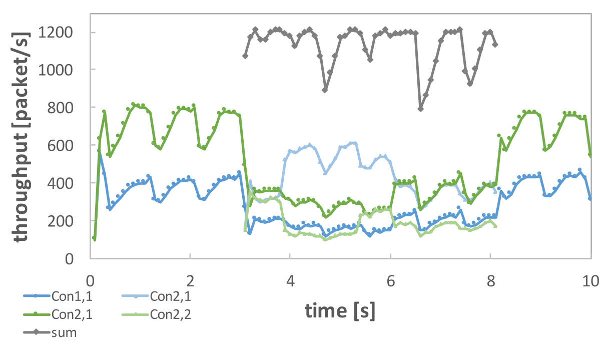
Fig. 12 Data throughputs in dumbbell topology when n = 1.

sum of $r_{c1,1}$ halved once and $r_{c2,1}$ halved once is lower than L_{n2} (= 1,200 packet/s). With simulation time, $r_{c1,1}$ becomes higher than $r_{c2,1}$ gradually. This occurs because $r_{c1,1}$ increases faster than $r_{c2,1}$. The reason is that the RTT between $Con_{1,1}$ and $Pro_{1,1}$ is shorter than that between $Con_{2,1}$ and $Pro_{2,1}$.

Figure 12 shows the Data throughputs at $Con_{1,1}$ and $Con_{2,1}$ in Stateful Forwarding and the proposed method. In Stateful



(a) Stateful Forwarding



(b) Proposed method

Fig. 13 Data throughputs in dumbbell topology when n = 2.

Forwarding, the Data throughputs at $Con_{1,1}$ and $Con_{2,1}$ change largely during the period from 3 to 8 seconds. The average Data throughput at $Con_{1,1}$ is 237 packet/s, and that at $Con_{2,1}$ is 432 packet/s. The sum of these Data throughputs is 669 packet/s, and so Stateful Forwarding achieves only the bottleneck link utilization of 55.8 percent. As shown in Fig. 12 (b), in the proposed method, the Data throughputs at $Con_{1,1}$ and $Con_{2,1}$ are more stable than in Stateful Forwarding. Moreover, during the period from 3 to 8 seconds, the average Data throughput at $Con_{1,1}$ is 540 packet/s, and that at $Con_{2,1}$ is 576 packet/s. The sum of the data throughputs is 1,116 packet/s, and so the proposed method achieves the bottleneck link utilization of 93.0 percent.

Figure 13 shows the Data throughput when n = 2. In Stateful Forwarding, the total Data throughput of four consumers changes largely during the period from 3 to 8 seconds. Its average is 643 packet/s and so Stateful Forwarding achieves only the bottleneck link utilization of 53.6 percent. On the other hand, in the proposed method, the total Data throughput of four consumers is more stable than in Stateful Forwarding. Its average is 1,137 packet/s and so the proposed method achieves the bottleneck link utilization of 94.7 percent. The Data throughputs of individual consumers differ largely in both methods.

Figure 14 summarizes the bottleneck link utilization during the period from 3 to 8 seconds when the number of consumers is 2, 4 and 8 (n = 1, 2 and 4). In any cases, the proposed method achieves higher values than Stateful forwarding. As for the fairness among individual consumers, the proposed method is better than Stateful Forwarding when n = 1. However, in the case of n = 2 and 4, the fairness is similarly poor in both methods. In order to improve the fairness, another mechanism needs to be introduced.

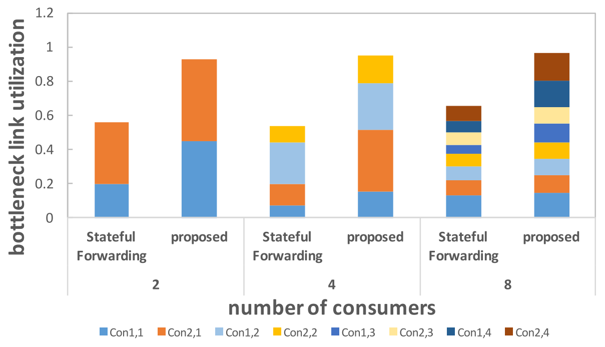


Fig. 14 Bottleneck link utilization with changing number of consumers.

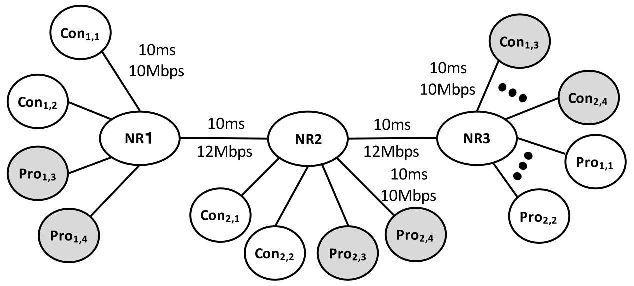


Fig. 15 Dumbbell topology with bidirectional content delivery.

4.2.3 Dumbbell Topology with Bidirectional Content Delivery

In the above evaluation, all Interest packets are transfered in the right direction and all Data packets are transfered in the left direction. In an actual environment, Interest and Data packets are exchanged in two directions. In order to evaluate the performance of bidirectional content delivery, we use a dumbbell topology shown in **Fig. 15**. In this configuration, four consumers ($Con_{1,1}$, $Con_{1,2}$, $Con_{2,1}$ and $Con_{2,2}$) are located in the left side, and other four consumers ($Con_{1,3}$, $Con_{1,4}$, $Con_{2,3}$ and $Con_{2,4}$) are located in the right side. Similarly, eight producers ($Pro_{1,1}$ through $Pro_{1,4}$ and $Pro_{2,1}$ through $Pro_{2,4}$) are located in both sides. One consumer downloads content from the corresponding producer. In the links between NR1 and NR2 and between NR2 and NR3, both Interest and Data packets are transfered. This configuration is the combination of the case when $n = 2$ in one direction (shown by white circles in Fig. 15), and the content delivery between four consumers in the right side and four producers in the left side (shown by grey circles in Fig. 15). We evaluate the bottleneck link utilization of each method, and compare the total Data throughputs in the unidirectional and bidirectional cases.

Figure 16 shows the bottleneck link utilization. In both consumers locations, the proposed method achieves higher utilizations than Stateful forwarding. In Stateful Forwarding, the utilization is 60.2 percent for the left side consumers and 67.1 percent for the right side consumers. In the proposed method, they are 81.7 percent and 89.8 percent, respectively. The fairness is also poor for both methods as in the unidirectional case.

Figure 17 shows the comparison between the total Data throughput in the unidirectional case with $n = 2$ and that of left side four consumers in the bidirectional case. Both in Stateful Forwarding and the proposed method, the peak value is less than 1,200 packet/s in the bidirectional case. This is because the bandwidth is used for Interest packets. In Stateful Forwarding, both

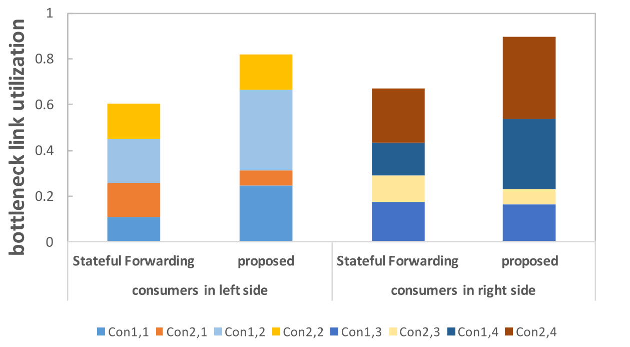
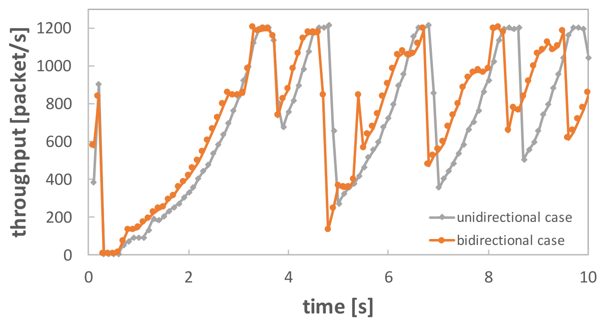
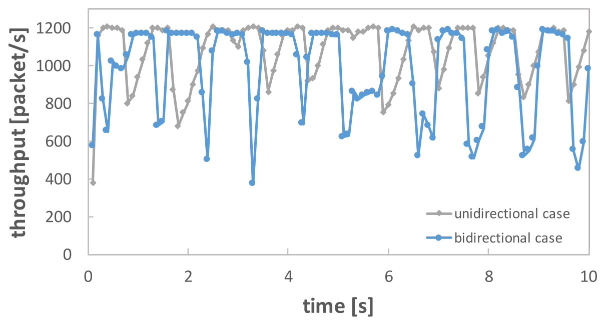


Fig. 16 Bottleneck link utilization when four consumers are located in either side.



(a) Stateful Forwarding



(b) Proposed method

Fig. 17 Comparison of total Data throughputs in unidirectional and bidirectional cases.

cases are similar and it is considered that the bandwidth sharing by Interest and Data packets gives little influence to the Data throughput. On the other hand, the total Data throughput in the bidirectional case becomes worse. The reason is considered that the duration stating in the *check* state is a little longer in the bidirectional case. This is because the propagation delay of Interest packets becomes large due to Data packet transfer.

5. Conclusion

In this paper, we have proposed a hop-by-hop rate control method in NDN. The proposed method suppresses the excessive reduction of the Interest sending rate due to the continuous NACK packets caused by the propagation delay. The proposed method limits the rate reduction to once per congestion. We have conducted a performance evaluation of the proposed method through computer simulations. We confirmed the effectiveness of the proposed method in both the linear topology and the dumbbell topology. In the future, we intend to consider the fairness of multiple flows in the proposed method. In addition, we will introduce the congestion avoidance procedure into the proposed method and conduct the performance evaluation of the proposed method in the topology considering in-network caching by intermediate routers.

Acknowledgments This work was supported by JSPS KAKENHI Grant Number 26280034.

References

[1] Xylomenos, G., Ververidis, C., Siris, V., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K. and Polyzos, G.: A Survey of Information-Centric Networking Research, *IEEE Communications Surveys & Tutorials*, Vol.16, No.2, pp.1024–1048 (2014).

[2] Vasilakos, A.V., Li, Z., Siris, V., Simon, G. and You, W.: Information Centric Network: Research challenges and opportunities, *Journal of Network and Computer Applications*, Vol.52, pp.1–10 (2015).

[3] Chen, Q., Xie, R. and Yu, F.R.: Transport Control Strategies in Named Data Networking: A Survey, *IEEE Communications Surveys & Tutorials*, Vol.18, No.3, pp.2052–2083 (2016).

[4] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N. and Braynard, R.: Networking named content, *Proc. ACM CoNext 2009*, pp.1–12 (2009).

[5] Ren, Y., Li, J., Shi, S., Li, L., Wang, G. and Zhang, B.: Congestion control in named data networking - A survey, *Computer Communications*, Vol.86, pp.1–11 (2016).

[6] Carofiglio, G., Gallo, M. and Muscariello, L.: ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking, *Proc. IEEE INFOCOM 2012*, pp.304–309 (2012).

[7] Salsano, S., Detti, A., Cancellieri, M., Pomposini, M. and Blefari-Melazzi, N.: Transport-layer issues in information centric networks, *Proc. ACM ICN2012* (2012).

[8] Saino, L., Cocora, C. and Pavlou, G.: CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content Centric Networking, *Proc. IEEE ICC 2013* (2013).

[9] Rozhnova, N. and Fdida, S.: An extended Hop-by-Hop Interest shaping mechanism for Content-Centric Networking, *Proc. IEEE GLOBECOM 2014* (2014).

[10] Fdida, S. and Rozhnova, N.: An effective hop-by-hop Interest shaping mechanism for CCN communications, *Proc. IEEE INFOCOM 2012*, pp.322–327 (2012).

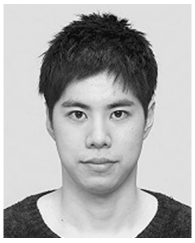
[11] Carofiglio, G., Gallo, M. and Muscariello, L.: Joint Hop-by-hop and Receiver-Driven Interest Control Protocol for Content-Centric Networks, *Proc. ACM ICN 2012*, pp.37–42 (2012).

[12] Cheng, Y., Afanasyev, A., Moiseenko, I., Zhang, B., Wang, L. and Zhang, L.: A case for stateful forwarding plane, *Computer Communications*, Vol.36, No.7, pp.779–791 (2013).

[13] Wang, Y., Rozhnova, N., Narayanan, A., Oran, D. and Rhee, I.: An Improved Hop-by-hop Interest Shaper for Congestion Control in Named Data Networking, *Proc. ACM ICN '13* (2013).

[14] Kato, T. and Bandi, M.: Congestion Control Avoiding Excessive Rate Reduction in Named Data Network, *Proc. IEEE CCNC 2017* (2017).

[15] Afanasyev, A., Moiseenko, I. and Zhang, L.: ndnSIM: NDN simulator for NS-3, NDN, Technical Report NDN-0005 (2012).



Takahiko Kato received his B.E. degree from Sophia University in 2016. He is currently a student in Graduate School of Science and Technology, Sophia University. His research interest is information centric networking. He is a member of IEICE.

Masaki Bandai received his B.E., M.E. and Ph.D. degrees from Keio University in 1996, 1998 and 2004, respectively. From 1998 to 2000, he joined Sony Corporation. From 2004 to 2010, he was an assistant professor of Shizuoka University. From 2006 to 2007, he was a visiting assistant professor of the University of British Columbia. He is currently an associate professor of Department of Information and Communication Sciences of Sophia University. His current research interests include computer network systems. He is a member of ACM, IEEE, IEICE and IPSJ.