Impacts of System Clock Granularity on Performance of NDN Rate-based Congestion Control

Toshihiko Kato, Kazuki Osada, Ryo Yamamoto, and Satoshi Ohzahata
Graduate School of Informatics and Engineering
University of Electro-Communications
Tokyo, Japan
e-mail: kato@is.uec.ac.jp, osada@net.is.uec.ac.jp, ryo_yamamoto@is.uec.ac.jp, ohzahata@is.uec.ac.jp

***Abstract*—Named Data Networking (NDN) is a widely adopted future Internet architecture that focuses on large scale content retrieval. The congestion control is one of the hot research topics in NDN, and the rate-based congestion control method is considered to be well suited. From the viewpoint of implementation, however, the rate-based method has an issue that it requires the fine-grained clock management, which is hard to implement in off-the-shelf computers. We focused this issue in our previous paper, and evaluated the performance in the case that consumers use a coarse-grained clock system. In this evaluation, we used the Stateful Forwarding as a target, which is a rate-based method proposed by the group proposing NDN. The simulation results showed that a coarse-grained clock system increases congestion. We also proposed a smooth Interest sending scheme under a coarse-grained clock system, which relieves congestion. However, our previous paper discussed only results with limited evaluation conditions, such as one consumer/producer pair configuration and a relatively low link speed. In this paper, we revisit the impact of system clock granularity of the performance of NDN rate based congestion control with practical evaluation conditions and with detailed analysis.**

Keywords- NDN; Congestion Control; Rate Control; Clock Management.

I. INTRODUCTION

This paper is an extension of our previous paper [1], which is presented in an IARIA conference.

Resulting from a drastic increase in Internet traffic forecast [2], there are many studies on the future Internet architecture called Information Centric Network (ICN), which is well suited for large scale content retrieval. Named Data Networking (NDN) [3] is a widely adopted platform for the ICN researches. The fundamental concept adopted in NDN is the name of required content, not the address of hosts containing the content. NDN uses two types of packets in all communications: an Interest packet and a Data packet. A user called a consumer that requests a specific content sends an Interest packet containing the content name. A server called a producer that provides the corresponding content data returns a Data packet to the consumer. NDN routers transferring the Data packet cache the packet for future redistribution [4].

The congestion control is one of the hot research topics in NDN [5]. Although it has been a hot topic in TCP, the mechanisms in TCP congestion control are limited to the congestion window management at data senders [6] and the simple explicit congestion notification at intermediate routers, which is recently introduced [7]. In contrast, various techniques can be introduced to the NDN congestion control. The receiver-driven window-based congestion control approach is similar to that in TCP. In this approach, congestion is detected by timeout [8][9] or the congestion notification [10], and the window for Interest packets are managed heuristically, e.g., through an Additive Increase and Multiplicative Decrease (AIMD) mechanism. In NDN, the rate-based congestion control approach is also studied actively. In this approach, a consumer and routers maintain a rate, in which Interest packets are transmitted contiguously. The rate is determined heuristically by use of congestion notification [11]-[13] or by the explicit rate reporting [14]-[16].

In NDN, the Round-Trip Time (RTT) between an Interest packet and the corresponding Data packet changes largely because of the Data packet caching at routers. The window-based congestion control approach needs to determine a window size corresponding to the delay and bandwidth product, but the delay changes in NDN. Therefore, it is pointed that the window-based approach is not suited to NDN and that the rate-based approach is more appropriate for NDN congestion control.

From the viewpoint of implementation, however, the rate-based congestion control approach has some problems. Since the transmission speed in recent data links becomes high, such as 1 Gbps, the fine-grained clock management is required in the rate-based congestion control. For example, if the Data packet size is 10,000 bits and the link speed is 1 Gbps, the interval of Interest packet transmission is 10 micro seconds (corresponding to 100 MHz) when Interest packets are transmitted in a line speed. If the rate is 0.5 Gbps or 0.3 Gbps, the Interest transmission interval will be 20 micro seconds (50 MHz) or 33.33 micro seconds (30 MHz), respectively. In order to handle these cases, it is supposed that higher precision clock with shorter tick, such as 1 micro second (1 GHz), will be required to control the Interest packet sending timing.

On the other hand, the fine-grained clock management is hard to implement in off-the-shelf computers. TCP implementation uses 200 msec and 500 msec clocks for the delayed acknowledgement and retransmission, respectively [17]. So, it is considered that implementing rate-based mechanism with micro second order clock is extremely hard.

We pointed out this issue and discussed how a coarse-grained clock system influences the NDN rate-based congestion control, in our previous paper [1]. We adopted the Stateful Forwarding [11] as a target system of evaluation,

because it is implemented in ndnSIM [18], which is a widely used network simulator of NDN. Moreover, we proposed a method to send Interest packets more smoothly even in the coarse-grained clock environment.

Although our previous paper gave some level of steady discussions and proposals, it has some problems in terms of the details of performance evaluation. The performance evaluation in our previous paper used a simple network configuration where one pair of consumer and producer connected via two routers using 10 Mbps links. The coarse-grained clock system used 50 msec through 200 msec tick intervals. This means that our previous paper provides only a trivial performance evaluation. In this paper, we revisit the issue of the impact of system clock granularity on the performance of NDN rate-based congestion control, with practical evaluation conditions. We add some evaluations on the maximum depth of token bucket used for rate control [19] in the evaluation described in our previous paper. We also provide some results of performance evaluation using a dumbbell network configuration with 100 Mbps links. The tick interval is 1 msec through 10 msec. Those evaluation results also show that the coarse-grained system clock gives some performance degradation of the rate-based congestion control and the proposed smoothening method improves the performance.

The rest of this paper is organized as follows. Section II explains the related work on NDN congestion control and discusses the system clock management. Section III describes the simulator-based performance evaluation of the Stateful Forwarding over a coarse-grained clock system. Section IV gives our proposal of smooth Interest packet sending even if the coarse-grained clock management is used. Section V provides the performance evaluation results using a dumbbell network configuration. In the end, Section VI concludes this paper.

II. Related Work

A. Related work on NDN congestion control

As described above, the congestion control methods in NDN are categorized as the window-based and the rate-based methods. The Interest Control Protocol (ICP) [8] and the Content Centric TCP (CCTCP) [9] are examples of the traditional TCP like window-based methods, where a consumer sends Interest packets with the limitation of window size, and the window size is changed according to the AIMD mechanism triggered by Data packet reception and congestion detected by timeout. The Chunk-switched Hop Pull Control Protocol (CHoPCoP) [10] is another window-based method. It introduces the explicit congestion notification with random early marking instead of the timeout-based congestion detection, and the Interest sending control is done at a consumer with the window size changing according to the AIMD mechanism. Although the window-based methods are simple, the window size itself may not be optimum when many Data packets are cached in different routers.

On the other hand, the rate-based methods are classified into the non-deterministic scheme, which uses the AIMD mechanism in determining the Interest sending rate, and the explicit rate notification scheme, in which intermediate routers report the optimum Interest rate to a consumer. The Stateful Forwarding (SF) [11] is an example of the former scheme. SF introduces a negative acknowledgment (NACK) packet, which has a similar packet structure with Interest, as a response to an Interest packet. NACK packets are generated when a router detects congestion. A consumer and a router manage the Interest sending rate locally by AIMD, and it decreases the rate when a NACK packet is received. The Stateful Forwarding with NACK suppressing [12] is a modification of SF. It resolves a problem that SF suffers from excessive rate reduction invoked by continuous NACK packets generated within one congestion event. The Practical Congestion Control (PCON) scheme [13] uses the CoDel active queue management scheme [20], which watches out the delay of packets in sending queues, to detecting congestion. When congestion is detected, a router signals it to consumers and downstream routers by explicitly marking Data packets. In respond to this reporting, the alternative path forwarding or the rate reducing is performed by downstream routers or consumers, respectively.

In contrast with those non-deterministic methods, new methods have emerged that enable routers to report a maximum allowed Interest sending rate. In the Explicit Congestion Notification (ECN) based Interest sending rate control method proposed in [14], a consumer uses a minimum rate among the reported rates from all intermediate routers. In the Hop-By-Hop Interest Shaping (HoBHIS) [15], routers decide the maximum allowed Interest sending rate independently and accordingly shape Interest packet. The maximum allowed rate is also reported to a consumer and this allow a consumer to send Interest packets without invoking congestion. The Multipath-aware ICN Rate-based Congestion Control (MIRCC) [16] introduces a similar per-link Interest shaper at every router and rate reporting to consumer. It takes account of the case that a flow uses multipath transfer. In those methods, the maximum allowed rate is calculated from the parameters including link capacity and utilization, queue size, inflated Interest rate, and average RTT. They are able to control Interest transmission so as to suppress congestion, and as a result they can provide higher throughput compared with other rate-based methods.

B. Discussions on clock management

Although the rate-based congestion control methods are capable to provide better performance than the window-based method, they have implementation issues. In order to control the timing to send Interest packets, timers need to be implemented that expire when Interest packets are sent out. If the link speed is high and there are a lot of content retrieval flows, the timeout values of those timers become small and the timeout timing will be random. In order to implement those timers over off-the-shelf computers, the fine-grained clock mechanism and multiple timers realized by timer interrupt handler are required. However, they will introduce large processing overhead and reduce processing throughput drastically.

In order to avoid this problem, TCP protocol processing uses very rough clock mechanism, as described above. The

Asynchronous Transfer Mode (ATM) [21], a legacy scheme standardized in the framework of broadband integrated services digital network, uses rate-based control for sending ATM cells. However, they do not use clock mechanism but adopt a way that null cells are inserted between user data cells in order to pace user data cell flows.

Yamamoto [22] tackled a similar issue for high speed TCP data transfer. He pointed out that the TCP over Gigabit link requires the rate control as well as the window control but the clock-based rate control provides large processing overhead for terminals. So, he introduced pause packets over Gigabit Ethernet, corresponding to null cells in ATM, that are used only between end nodes and switching hubs. This method can be adopted only over the dedicated link and cannot be applied to the shared media type link like high speed wireless LAN.

Kato and Bandai mentioned a similar issue on the processing overhead of fine-grained clock management for the rate-based congestion control, but they took a method that exploits a hop-by-hop window control [23].

III. FUNDAMENTAL PERFORMANCE EVALUATION WITH COARSE-GRAINED CLOCK

Based on the discussions in Section II.B, we evaluate how the rate-based NDN congestion control works when the clock granularity is coarse. We adopt SF [11] as a target rate-base scheme because it is implemented by its proposer over ndnSIM version 1.0 [18], which uses C++ as a programming language. This section discusses the fundamental performance evaluation when the clock management becomes coarse-grained.

A. Software implementation

Currently, ndnSIM has several versions; 1.0, and 2.0 through 2.6. Although SF is proposed by the research group who is maintaining ndnSIM, we believe that SF is implemented only in ndnSIM 1.0. Moreover, there are some bugs and problems in ndnSIM 1.0. For evaluating the influence by coarse-grained clock system, we added the followings to the current ndnSIM software.

- Support of AIMD like rate control

  SF mentions the rate control using AIMD as one possible candidate, but ndnSIM does not implement it. So, we implemented it in the module managing Interest and Data packets (the `ForwardingStrategy` class) in the following way. The start value of Interest sending rate is given manually. When a router receives a Data packet, it increases the rate by one, under the limitation that it does not exceed the link speed at the outgoing interface. When receiving a NACK packet, it halves the current rate, under the limitation that the minimum value of Interest sending rate is 1 packet/s.

  It should be noted that the intermediate routers do not provide a shaping function that transmits Interest packets in a fixed rate. Instead, it provides a policing function that checks whether the Interest sending rate exceeds the limit or not. In order to handle a variable sending rate, the policing is performed by use of a token bucket as described above.

- Use of constant bit rate (CBR) type consumer

  ndnSIM 1.0 provides three types of consumers: rate-based (the `ConsumerCbr` class), window-based (the `ConsumerWindow` class) and batch-type (the `ConsumerBatches` class). We decided to use the `ConsumerCbr` class and added the AIMD like rate control on it. This class uses a protected static variable `m_frequency` as the Interest sending rate. We changed the variable in the same way described above in the `OnData()` and `OnNack()` methods, which are the methods called when a Data packet and a NACK packet is received, respectively.

- Emulation of coarse-grained clock system

  In NDN, the rate control is implemented in the classes `Consumer` and `ConsumerCbr`; the `Consumer` class is the superclass of `ConsumerCbr`. The sending of Interest packets with a specific rate is implemented in the `ScheduleNextPacket()` method of the `ConsumerCbr` class. In this method, the `SendPacket()` method of the `Consumer` class is invoked periodically, every `1.0/m_frequency` seconds. The `SendPacket()` method sends one Interest packet actually.

  We emulated a course-grained clock system in the `Consumer` class in the following way (see Figure 1).

  - A clock system with longer tick, such as 100 msec, is implemented in the `Consumer` class. It calls itself periodically with the `Schedule()` method of the `Simulator` class.
  - We also introduced a queue storing Interest packets temporarily. This queue is implemented using the `list` class.
  - In the `SendPacket()` method, Interest packets are stored in the queue, instead of being sent actually.
  - When the longer clock tick is invoked, all the queued Interest packets are transmitted actually.

- Specifying bucket maximum depth explicitly

  In ndnSIM 1.0, a token bucket is implemented in the `LimitsRate` class. We introduce a constant which manages the maximum depth of the bucket.

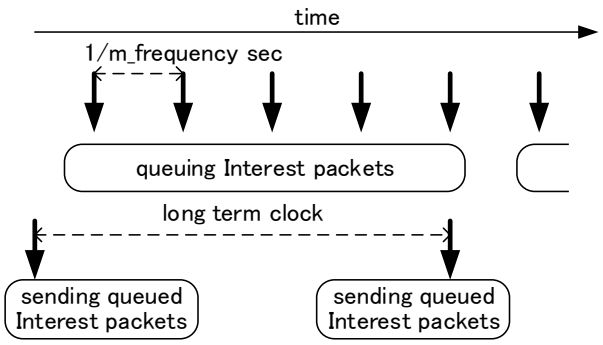


Figure 1. Implementation scheme of coarse-grained clock system.

B. *Performace evaluation with simple network*

As a fundamental performance evaluation, we conducted the evaluation described in this section.

(1) Experimental setting

The network configuration used in this evaluation is shown in Figure 2, which is a linear configuration where one consumer (C), two routers (R1 and R2), and one producer (P) are connected via 10 Mbps link with 50 msec propagation delay. The length of a Data packet is 1250 bytes, and the link speed corresponds 1,000 packets/sec. As described above, a consumer and routers maintain a token bucket for policing the Interest packet flow. The arriving Interest packet is thrown into the token bucket conceptually, and, if the depth of the bucket becomes larger than the maximum value, a NACK packet is replied for the Interest packet. In our experiment, the maximum depth is set to 50 packets.

Under these conditions, we evaluated the cases that the coarse-grained clock has 50 msec, 100 msec, and 200 msec tick intervals. In all the evaluation runs, the consumer starts from 200 packets/sec as the Interest sending rate. Each evaluation run takes 10 sec.

Figure 3 shows the time variation of the sequence number contained in the name of requested content. It corresponds to the number of content request in a content retrieval flow. Each value is plotted when the corresponding Interest packet is sent. Figure 4 shows the time variation of the Interest sending rate at the consumer. In this figure, each value is plotted when the consumer receives a Data or NACK packet and it changes the value of Interest sending rate.

The orange lines in Figures 3 and 4 show the results of the original SF implementation. The sequence number is increasing steadily. The Interest sending rate starts from 200 packets/sec and goes to 1,000 packets/sec, the maximum value corresponding to the link speed. These results show that the rate-based congestion control works well.

The gray line in Figures 3 and 4 show the results when the coarse-grained clock system is used and the tick interval is 50 msec. The sequence number is also increasing steadily, but there are several drops in the Interest sending rate. The rate starts from 200 packets/sec and goes to 1,000 packets/sec, but it drops to 500 packets/sec at 3.2 sec. This is triggered by a NACK packet generated locally inside the consumer. That is, the consumer also maintains the token bucket for policing the Interest packet flow. When the Interest sending rate is 1,000 packets/sec and the tick interval is 50 msec, fifty Interest packets are generated in one moment by the application, and rush into the bucket. Since the maximum depth of the bucket is 50 packets, all of them are stored in the bucket and leaked in 1,000 packets/sec (actually they are transmitted to R1 in a line speed). But in some timing, fifty Interest packets are

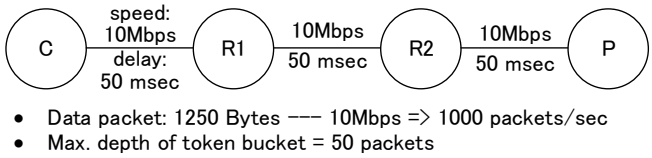


Figure 2. Network configuration and conditions in fundamental evaluation.

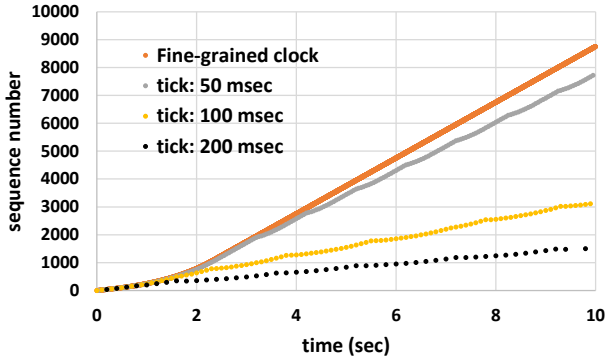


Figure 3. Time variation of Interest sequence number.

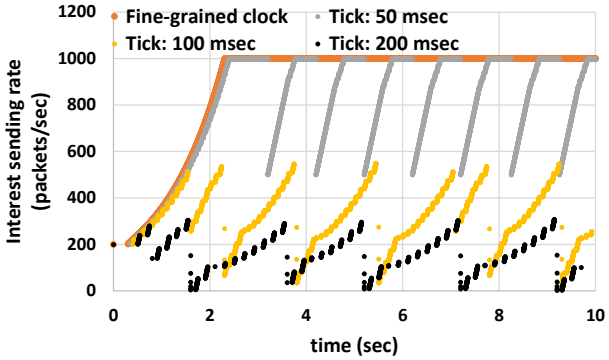


Figure 4. Time variation of Interest sending rate.

generated in the situation that there are some packets remaining in the bucket. Then, a NACK packet is generated.

The yellow lines and the black lines in Figures 3 and 4 show the results when the tick interval is 100 msec and 200 msec, respectively. In these cases, the increase of the sequence number is suppressed, and the Interest sending rate is limited up to 600 and 300 packets/sec, respectively. This is because the number of Interest packets transmitted back to back is increasing. These results show that, when the tick interval becomes large in the coarse-grained clock system, the rate-based congestion control does not work correctly.

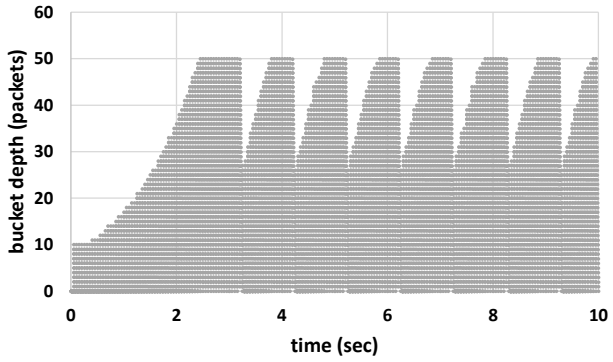
Table I gives a summary of the results. The Data packet throughput is the total content size transferred during an evaluation run divided by ten seconds. In the case of the fine-grained clock (Original in the table), the throughput is 8.75 Mbps and there are no NACK packets transferred. In the case of the coarse-grained clock with 50 msec tick, the Data packet throughput decreases slightly, because the rate goes to 1,000 packets/sec and there are no contiguous NACK receiving. However, the cases with 100 msec tick and 200 msec tick, the

TABLE I. SUMMARY OF RESULTS WITH COARSE-GRAINED CLOCK.

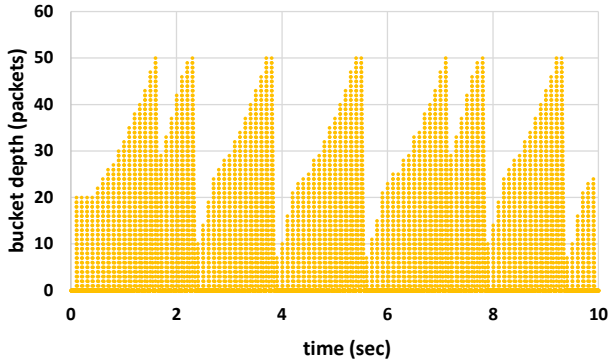
	Original	Tick = 50 msec	Tick = 100 msec	Tick = 200 msec
Data packet throughput (Mbps)	8.75	7.72	3.12	1.50
Number of NACK packets	0	7	20	27

number of NACK packets increases and the Data packet throughput decreases largely.

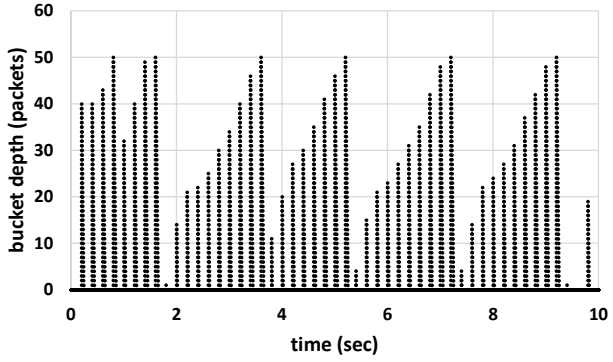
We also investigated how the token bucket depth changes. Figure 5 shows the time variation of the token bucket depth at the consumer. Figure 5 (a) is the result for the tick interval of 50 msec. In this case, the bucket depth increases up to 50 packets, which is the maximum depth, and then it keeps the value for around 0.5 sec. In the case that the tick interval is 50 msec, fifty Interest packets are transmitted in a group when the rate is 1,000 packets/s, the maximum value corresponding to the line speed. This is the same as the maximum bucket depth. Therefore, a group of Interest packets transmitted in the line speed pile fifty tokens in the bucket, which are released from the bucket just before the next group are generated. This procedure is repeated for around 0.5 sec, and in some timing, a token exceeds the maximum depth. This generates a NACK packet and the Interest sending rate is halved.



(a) tick = 50 msec.



(b) tick = 100 msec.



(c) tick = 200 msec.

Figure 5. Time variation of token bucket depth in consumer.

Figure 5 (b) shows the result for the tick interval of 100 msec. In this case, it is possible to send up to 100 Interest packet in a group, but when the rate becomes 510 packet/s, the Interest packet burst contains fifty one packets and the bucket overflows. Since multiple NACK packets are generated, the rate is reduced to 1 packet/s.

Figure 5 (c) shows the result for the tick interval of 200 msec. In this case, when the rate becomes 255 packets/s, the generated Interest burst will make the bucket overflow. Although the frequency of the bucket overflow is similar with the case of 100 msec tick, the throughput will be lower since the number of Interest packets sent is smaller than the case of 100 msec tick.

IV. Proposal to Smoothen Interst Packet Sending

A. Proposed method

In the SF mechanism with the coarse-grained clock system described in Section III, we supposed that Interest packets are transmitted only in response to ticks. As a result, Interest packets were sent in a burst and this triggered the overflow in the token bucket.

Here, we propose an Interest control method that utilizes the Data and NACK packet receiving timing. When a consumer receives a Data or a NACK packet, the receiving processing is triggered by a hardware interrupt mechanism, and it does not give large overhead to computers, different from the software based timeout mechanism. So, the receiving timing is a good chance to proceed the Interest packet sending. So, we have added the following mechanism in the coarse-grained clock system described in Section III.A.

- When a consumer receives a Data or a NACK packet, it processes the received packet and then tries to send the Interest packets stored in the Interest queue.
- This procedure is implemented in the `OnData()` and `OnNack()` methods in the `Consumer` class.

B. Perforamce evaluation results in simple configuration

We have conducted the performance evaluation of the proposed method in the same configuration and conditions as the previous section. Figure 6 shows the time variation of the Interest sending rate at the consumer implementing the proposed method.

Different from the results given in Figure 4, all the cases when the tick interval is 50 msec, 100 msec, and 200 msec give the similar results with the fine-grained clock system. That is, the Interest sending rate starts from 200 packets/sec, goes to 1,000 packets/sec straightly, and keeps in this level. This means that there are no NACK packets generated. These results mean that the proposed method is effective for smoothening the bursty Interest packet sending caused by the coarse-grained clock system.

Table II shows a summary of the results. There are no NACK packets in all the cases of three tick interval values.

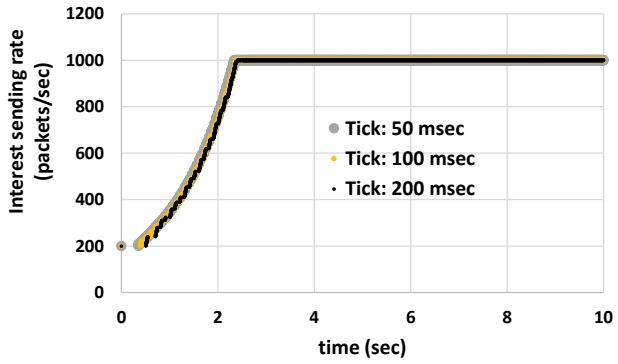


Figure 6. Time variation of Interest sending rate in proposed method.

TABLE II. SUMMARY OF RESULTS WITH PROPOSED METHOD.

	Tick = 50 msec	Tick = 100 msec	Tick = 200 msec
Data packet throughput (Mbps)	8.73	8.70	8.69
Number of NACK packets	0	0	0

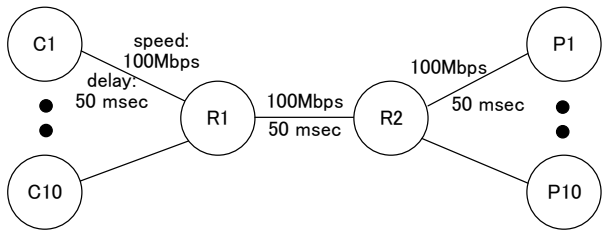
The Data throughput are also similar for three cases, and the value is close to that of the fine-grained clock based SF.

V. PERFORMANCE EVALUATION WITH REALISTIC ENVIRONMENTS

The performance evaluation conditions described in Sections III and IV are too simple because of linear network configuration, relatively low link speed and relatively long tick intervals. In this section, we provide the results of performance evaluation in more practical conditions.

A. Evaluation conditions

Figure 7 shows the network configuration used by the performance evaluation described in this section. It is a dumbbell network. Ten consumers (C1 through C10) are connected to router R1, which is connected to another router R2. Ten producers (R1 through R10) are connected to router R2. All the links have the link speed of 100 Mbps and the one way transmission delay of 50 msec. The size of a Data packet is 1,250 bytes, i.e. 1Kbit. The line speed corresponds to 10 Kpackets/sec in terms of Data packets. The detailed evaluation conditions are as follows.



- Data packet: 1250 Bytes --- 100Mbps => 10 Kpackets/sec
- Max. depth of token bucket = 50 packets, 100 packets, 500 packets
- Tick interval = 1 msec, 5 msec, 10 msec

Figure 7. Network configuration and conditions in practical evaluation.

- One consumer is supposed to retrieve a dedicated content from the corresponding producer, e.g., C1 from R1. This means that no Data packet caching is used.
- In the evaluation, we changed the number of consumer/producer pairs from 1 to 10. In the case of the number of pairs is one, it is a linear network configuration. In this case, the token bucket in the consumer will overflow, as in the evaluation described in Section III. When it is more than one, the link between R1 and R2 becomes the bottleneck link and the token bucket in router R1 will overflow.
- The initial value of Interest sending rate is set to the maximum rate (10 Kpackets/sec) divided by the number of consumer/producer pairs. For example, when the number of pairs is two, the initial Interest sending rate is 5 Kpackets/sec for two consumers.
- The sending of Interest packets will start at the same time among the consumers. That is, the Interest sending will be synchronized at least in the beginning of the evaluation runs. This is a very heavy condition, but, since the token buckets are prepared for individual consumer/producer pairs, the overflow will occur independently and at different timings for different pairs. So, the impact of this condition seems to be not large.
- The maximum value of token bucket depth is set to 50 packets, 100 packets, or 500 packets. Since the congestion is invoked by the overflow at the token bucket, we tested for different maximum values.
- The tick interval vales are selected from 1 msec, 5 msec, and 10 msec. We believe that these values are reasonably small to be implemented in off-the-shelf computers.

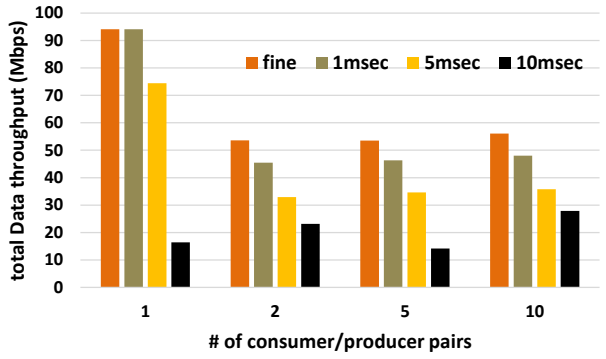
In the case of the evaluations in Sections III and IV, there was only one Interest packet flow. So, we used the Interest sending rate limit assigned for individual outgoing interface. This is called `PerOutFaceLimits` implemented in the `LimitsRate` class in ndnSIM 1.0. In the case of this evaluation, however, the rate limit needs to be prepared for individual Interest flow as well as for outgoing interface. This is realized by `PerFibLimits` implemented in the same class. It should be mentioned that there is a bug for `PerFibLimits` in ndnSIM 1.0. In the method `NotifyNewAggregate()` in the `LimitsRate` class, which is called just after the instance is created, the `LeakBucket ()` method needs to be scheduled in order to start the periodical token release. However, this is done only for `PerOutFaceLimits`, and not for `PerOutFaceLimits`. So, we modified this part of program in ndnSIM 1.0.

B. Performance evaluation results of SF under coarse-grained clock

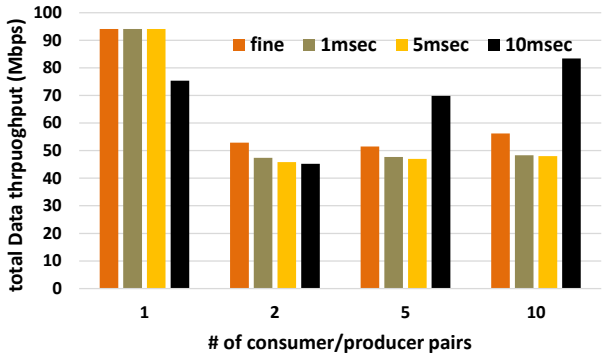
We conducted performance evaluation of SF under the conditions described above. The execution time of one

evaluation run is 10 sec similarly with the evaluation described in Sections III and IV. Figure 8 shows the results of Data packet throughput. The graph indicates the sum of Data packet throughput of individual consumer/producer pairs. So, the limit is 100 Mbps. The horizontal axis of the graph is the number of consumer/producer pairs; 1, 2, 5, and 10. The figure shows the three cases with different maximum values of token bucket depth; 50 packets, 100 packets, and 500 packets. Figure 9 shows the total number of NACK packets generated in individual consumer/producer pairs. We can discuss the following from those results.

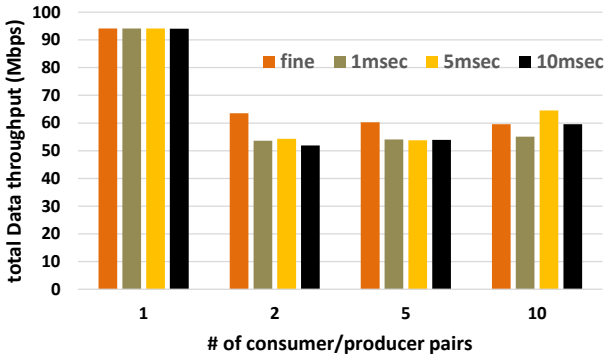
- Figure 8 shows that, when there are multiple Interest/ Data packet flows, the total Data packet throughput decreases. This is because SF itself has some performance problem in the case of multiple flows [12]. In the dumbbell network, the Interest sending rate of individual flow increases to the maximum value corresponding to the line speed between a consumer and the bottleneck router. This triggers network congestion at the bottleneck router, and generates a number of NACK packets. So, the Data throughput degradation and the increase of NACK packets in the case of multiple flows come from factors other than the coarse-grained system clock.
- In the case of a single Interest/Data flow between one pair of consumer/producer, the coarse-grained clock degrades the Data throughput. When the maximum bucket depth is 50 packets (Figure 8 (a)), the tick



(a) max. bucket depth = 50 packets.

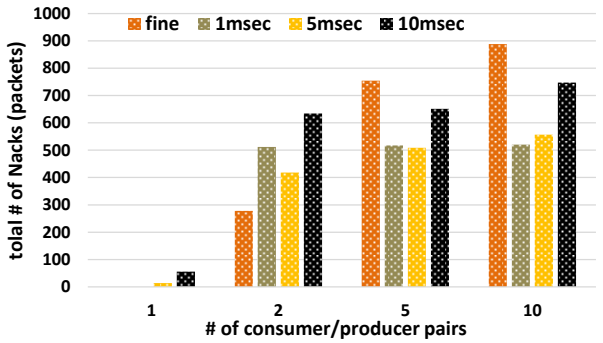


(b) max. bucket depth = 100 packets.

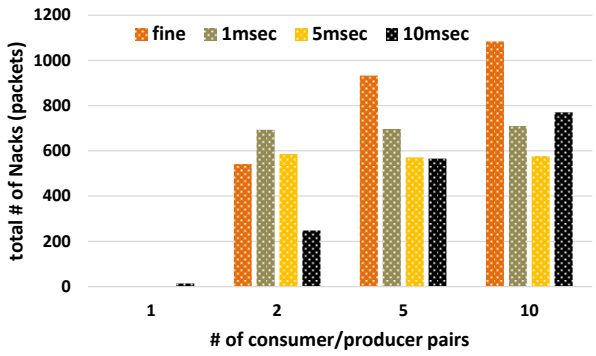


(c) max. bucket depth = 500 packets.

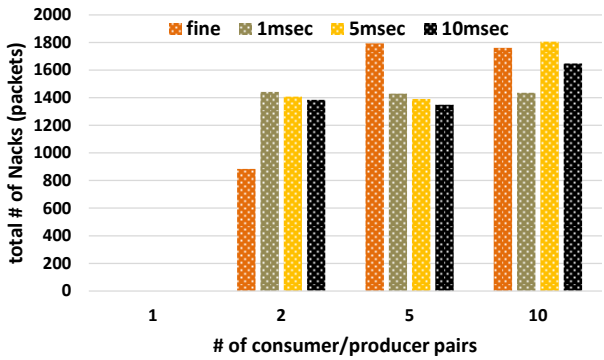
Figure 8. Data throughput under practical evaluation conditions.



(a) max. bucket depth = 50 packets.



(b) max. bucket depth = 100 packets.



(c) max. bucket depth = 500 packets.

Figure 9. Number of NACK packets under practical evaluation conditions.

interval of 5 msec degrades the throughput slightly, and the tick interval 10 msec induces a heavy throughput degradation. When the maximum bucket depth is 100 packets, a slight throughput degradation is observed for the tick interval of 10 msec. There are no throughput degradations for a single flow when the maximum bucket depth is 500 packets.

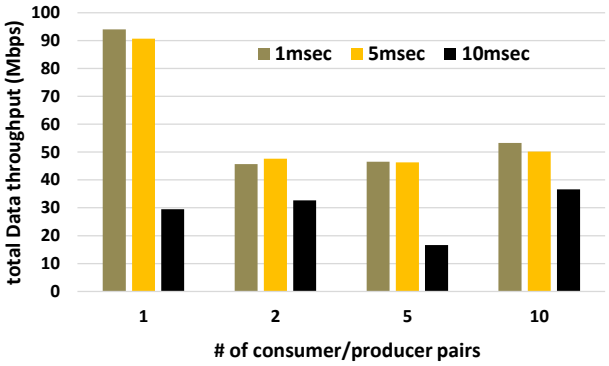
- When there are multiple Interest/Data flows, the affects by the coarse-grained clock is clearly given in the case that the maximum bucket depth is 50 packets (Figure 8 (a)). According to the tick interval's increasing, the total Data packet throughput is decreasing. In the case that the maximum bucket depth is 100 packets or 500 packets (Figure 8 (b) or 8 (c)), the Data throughput of SF with fine-grained clock is higher than that under coarse-grained clock, except the case of tick interval = 10 msec and maximum bucket depth = 100 packets.
- As for the fairness among multiple Interest/Data flows, Figure 10 shows the Data packet throughput of individual flows in the case of tick interval = 1 msec and maximum bucket depth = 50 packets. For individual cases of consumer/producer pairs, the Data throughput of individual flow is similar with each other.

C. *Performance evaluation results of SF with smoothening Inerest packet sending under coarse-grained clock*

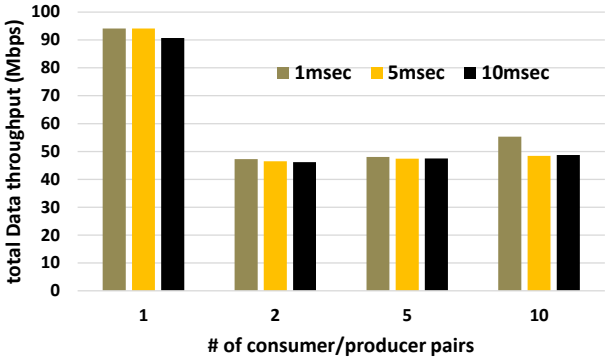
Figure 11 shows the evaluation results when the proposed smoothening Interest sending rate method is used under the coarse-grained clock. In this case, the results are different for the case of single Interest/Data flow and for the case of multiple flows.

When there is only a single flow, the total Data packet throughput is improved by the proposed method. For example, in the case of tick interval = 5 msec and maximum bucket depth = 50 packets, the total Data throughput increased to 91 Mbps from 74 Mbps. In the case of tick interval = 10 msec and maximum bucket depth = 50 packets, the throughput becomes 30 Mbps, which was 16 Mbps without the smoothening method. In the case of tick interval = 10 msec and maximum bucket depth = 100 packets, the throughput increased to 91 Mbps from 75 Mbps. That is, it can be concluded that in the single Interest/Data flow case, the proposed smoothening method is effective to increase Data throughput when a coarse-grained clock is adopted.

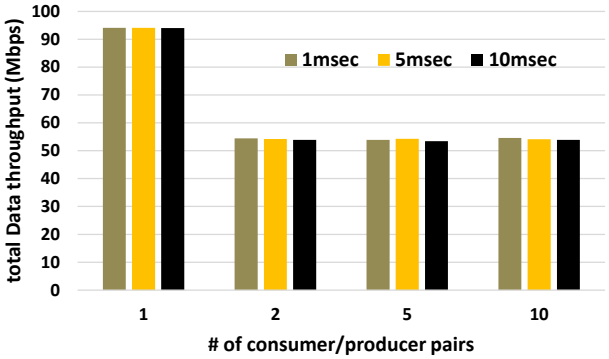
Figure 10. Individual Data packet throughput at tick interval = 1 msec and max. bucket depth = 50 packets.



(a) max. bucket depth = 50 packets.



(b) max. bucket depth = 100 packets.



(c) max. bucket depth = 500 packets.

Figure 11. Data throughput with Interest sending smoothening.

However, when there are multiple Interest/Data flows, the proposed method does not improve the total Data packet throughput. The throughput in Figure 11 is similar with the corresponding results in Figure 8. The reason for this result will be that SF has other performance problems, such as the excessive rate reduction by continuously transmitted NACK packets [12], and they cancel the effects of the Interest sending smoothening.

VI. CONCLUSIONS

This paper revisited an issue on how the coarse-grained clock system influences the NDN rate-based congestion control, which was pointed out in our previous paper [1]. Currently, the rate-based congestion control is considered to be effective in NDN. However, the rate-based control over high speed links requires highly precious clock management and this gives a serious processing overhead to off-the-shelf computers. So, we think that commodity based consumers need to use a coarse-grained clock system.

In the fundamental performance evaluation using Stateful Forwarding as a target system, we showed the following. Even if the network does not cause any congestion, the tick intervals such as 50 msec, 100 msec, and 200 msec generate some NACK packets. Especially, in the cases of 100 msec and 200 msec ticks, the Data throughput decreases largely. These results mean the NDN rate-based congestion control has some problems when it is used with a coarse-grained clock system.

This paper also proposed a scheme to smoothen Interest sending, which allows a queued Interest packets for sending to be transmitted when any Data or NACK packets are received. As the result of fundamental simulation evaluation, the proposed method did not generate any NACK packets even if 50 msec, 100 msec, and 200 msec are used as tick intervals.

This paper also showed the evaluation results using more practical network configuration, with higher link speed, multiple Interest/Data flows, and shorter tick intervals such as 1msec through 10 msec. In this evaluation, the results were a little different in a single flow case and a multiple flow case. When there is only one Interest/Data flow, the coarse-grained clock induced the Data packet throughput, even the tick interval is 1 msec. The proposed smoothening method also recovered the Data throughput, as in the fundamental evaluation.

However, when there are multiple Interest/Data flows, the situation changed. In this case, the link between intermediate routers in a dumbbell network becomes the bottleneck. Stateful Forwarding itself degrades the Data throughput in this case. Although the coarse-grained clock degrade the throughput in some conditions, the issue of Stateful Forwarding has a larger impact. The proposed smoothening method did not increase the throughput, either.

Recently, several rate control methods with explicit rate reporting are proposed. They will resolve the non-deterministic rate selection which Stateful Forwarding relies on. It is considered that we need to apply our methodology to those new types of rate-based congestion control methods in the future.

REFERENCES

[1] T. Kato, K. Osada, R. Yamamoto, and S. Ohzahata, "A Study on How Coarse-grained Clock System Influences NDN Rate-based Congestion Control," Proc. of IARIA ICN 2018, pp. 35-40, Apr. 2018.

[2] Cisco public, "Cisco Visual Networking Index: Forecast and Methodology, 2016-2021," Jun. 2017 [retrieved: Sep. 2018].

[3] V. Jacobson et al., "Networking Named Content," Proc. of CoNEXT '09, pp. 1-12, Dec. 2009.

[4] N. Minh, R. Yamamoto, S. Ohzahata, and T. Kato, "A Routing Protocol Proposal for NDN Based Ad Hoc Networks Combining Proactive and Reactive Routing Mechanism," Proc. of IARIA AICT 2017, pp. 80-86, Jun. 2017.

[5] Y. Ren, J. Li, S. Shi, L. Li, G. Wang, and B. Zhang, "Congestion control in named data networking - A survey," Computer Communications, vol. 86, pp. 1-11, Jul. 2016.

[6] A. Afanasyev, et al., "Host-to-Host Congestion Control for TCP," IEEE Commun. Surveys & Tutorials, vol. 12, no. 3, pp. 304-342, 2010.

[7] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," IETF RFC 3168, Sep. 2001.

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

[9] L. Saino, C. Cocora, and G. Pavlou, "CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content Centric Networking," Proc. of IEEE ICC 2013, pp. 3775-3780, Jun. 2013.

[10] F. Zhang, Y. Zhang, A. Reznik, H. Liu, C. Qian, and C. Xu, "A Transport Protocol for Content-Centric Networking with Explicit Congestion Control," Proc. of IEEE ICCCN 2014, pp. 1-8, Aug. 2014.

[11] Y. Cheng, A. Afanasyev, I. Moiseenko, B. Zhang, L. Wang, and L. Zhang, "A case for stateful forwarding plane," Computer Communications, vol. 36, no. 7, pp. 779-791, Apr. 2013.

[12] T. Kato and M. Bandai, "Congestion Control Avoiding Excessive Rate Reduction in Named Data Network," Proc. of IEEE CCNC, pp. 1-6, Jan. 2017.

[13] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A Practical Congestion Control Scheme for Named Data Networking," Proc. of ACM ICN 2016, pp. 21-30, Sep. 2016.

[14] J. Zhang, Q. Wu, Z. Li, M. A. Kaafar, and G. Xie, "A Proactive Transport Mechanism with Explicit Congestion Notification for NDN," Proc. of IEEE ICC 2015, pp. 5242-5247, Jun. 2015.

[15] N. Rozhnova and S. Fdida, "An extended Hop-by-Hop Interest shaping mechanism for Content-Centric Networking," Proc. of IEEE GLOBECOM 2014, pp. 1-7, Dec. 2014.

[16] M. Mahdian, S. Arianfar, J. Gibson, and D. Oran, "Multipath-aware ICN Rate-based Congestion Control," Proc. of ACM ICN 2016, pp. 1-10, Sep. 2016.

[17] K. Fall and W. Stevens, "TCP/IP Illustrated, Volume1; The Protocols, Second Edition," Addison-Wesley,

[18] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," NDN, Technical Report NDN-0005, 2012, Oct. 2012.

[19] "Overall ndnSIM documentation; Forwarding Strategies," http://ndnsim.net/1.0/fw.html [retrieved: Sep. 2018].

[20] K. Nichols and V. Jacobson, "Controlling Queue Delay," ACM Magazine Queue, vol. 10, issue 5, pp. 1-15, May 2012.

[21] ITU-T, "B-ISDN asynchronous transfer mode functional characteristics," Series I: Integrated Services Digital Nework, Recommendation I.150, Feb. 1999.

[22] Y. Yamamoto, "Estimation of the advanced TCP/IP algorithms for long sistance collaboration," Fusion Engineering and Design, vol. 83, issue 2-3, pp. 516-519, Apr. 2008.

[23] T. Kato and M. Bandai, "A Congestion Control Method for NDN Using Hop-by-hop Window Management," Proc. of IEEE CCNC 2018, pp. 1-6, Jan. 2018.