Proceedings of the Federated Conference on Computer Science and Information Systems pp. 479–488
DOI: 10.15439/2020F40
ISSN 2300-5963 ACSIS, Vol. 21

How Coarse-grained Clock Impacts on Performance of NDN Rate-based Congestion Control with Explicit Rate Reporting

Toshihiko Kato, Takumi Enda, Ryo Yamamoto and Satoshi Ohzahata
Graduate School of Informatics and Engineering, University of Electro-Communications
1-5-1, Chofugaoka, Chofu, Tokyo 182-8585 Japan
Email: {kato, enda, ryo_yamamoto, ohzahata}@net.lab.uec.ac.jp

***Abstract*—Named Data Networking (NDN) is a widely adopted future Internet architecture well-suited to large scale content retrieval. The congestion control is one of the research topics actively studied, and the rate-based congestion control method is considered to be fitted to NDN. 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. Among the rate-based congestion control methods, an approach in which intermediate nodes report a maximum rate explicitly for a flow is considered to work well. In this paper, we pick up the Multipath-aware ICN Rate-based Congestion Control as an example of explicit rate reporting scheme, and examine how coarse-grained clock gives impacts to its performance. This paper provides the performance evaluation when consumers and NDN routers use the system clock with long time interval. This paper also proposes a method for smoothening Interest sending under a coarse-grained clock and evaluates the performance of proposed method.**

I. Introduction

RESULTING from a drastic increase of content retrieval traffic over the Internet [1], there are many studies on the future Internet architecture called Information Centric Network (ICN). Among them, Named Data Networking (NDN) [2] is a platform widely adopted among 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 *consumer* that requests a specific content sends an Interest packet containing the content name. A *producer* that provides the corresponding content data returns a Data packet to the consumer. An Interest packet is forwarded using the name prefix it contains, and a Data packet traverse the reverse path of the corresponding Interest packet. *NDN routers* transferring a Data packet cache the packet itself for future redistribution.

The congestion control is one of the hot research topics in NDN [3]. Although it has been also a hot topic in TCP, the mechanisms in TCP congestion control are limited to the congestion window management at data senders [4] and the simple explicit congestion notification [5]. In contrast, various techniques are adopted to the NDN congestion control.

The receiver-driven window-based congestion control approach in NDN is similar to that in TCP. In this approach, the window for Interest packets is maintained in consumers, and Interest packets are sent back to back within the window size. In the traditional proposals, congestion is detected by timeout [6], [7] or the congestion notification [8], and the window size is managed heuristically through an Additive Increase and Multiplicative Decrease (AIMD) mechanism. There are some newly introduced methods, including one which adopts CUBIC TCP like window management [9], ones which use the active queue management scheme such as CoDel [10], watching out the delay of packets in sending queues for congestion detection [11], [12], and one adopts multiple window increase methods dependent on the usage of communication links [13].

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 regularly with a fixed interval. The rate is determined heuristically by use of congestion notification [14], [15] or by the explicit rate reporting [16]-[20]. Among these methods, the rate-based method with explicit rate reporting provides the best performance. Here, each router monitors the total of data packet traffic receiving over an individual link to an upstream router or a producer, and calculates the optimal Interest packet rate for each Interest-Data flow. Each router sends this optimal rate in a Data packet, and a consumer sends Interest packets according to the reported rate.

From the viewpoint of implementation, however, the rate-based congestion control approach has a problem. Since the transmission speed in recent data links becomes high, e.g., 1 Gbps for typical access links, 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 μsec 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 μsec or 33.33 μsec, respectively. In order to handle these cases, it is supposed that higher precision clock with shorter tick, such as 1 μsec, will be required in order to manage the Interest packet sending timing.

On the other hand, it is considered that the fine-grained clock management is hard to implement in off-the-shelf computers. In the real world, TCP implementation uses 200 msec (5Hz) and 500 msec (2Hz) clocks for the delayed acknowledgement and retransmission, respectively [21]. So, implementing a rate-based mechanism with micro second order clock is extremely hard, especially in NDN nodes handling a large number of flows simultaneously.

We pointed out this issue and discussed how a coarse-grained clock influences the NDN rate-based congestion control in our previous paper [22]. We adopted the Stateful Forwarding [14] as a target system of the evaluation, and showed that the performance, specifically the Data packet throughput, is degraded largely when a coarse-grained clock is introduced.

However, the Stateful Forwarding is not the best example of the NDN rate-based congestion control methods. As stated above, the explicit rate reporting methods, especially the Multipath-aware ICN Rate-based Congestion Control (MIRCC), provide better performance. In this paper, we examine how the coarse-grained clock influences the performance of MIRCC. We gave some study on this topic in our recent paper [23] but the analysis in this paper is insufficient. Moreover, we propose a method to send Interest packets more smoothly even in the coarse-grained clock environment.

It should be noted that [24] focused on a similar issue on the processing overhead of fine-grained clock management for the rate-based congestion control, but it took a method that exploits a hop-by-hop window control, which does not require the clock management.

The rest of this paper is organized as follows. Section 2 gives the related work focusing on the overview of NDN congestion control, the overhead of fine-grained clock, and MIRCC. Section 3 describes the implementation of MIRCC over the ndnSIM simulator [25], which is a widely used network simulator for NDN, and the modification under the coarse-grained clock. Section 4 gives the performance evaluation results of the original MIRCC with coarse-grained clock. Section 5 proposes a mechanism for smoothening the Interest sending under coarse-grained clock and shows its performance evaluation. In the end, Section 6 concludes this paper.

II. Related Work

A. Related work on NDN congestion control

As described above, most of the congestion control methods in NDN are classified into the receiver-driven window-based and the rate-based methods. Tables 1 and 2 show examples of those methods.

As examples of the traditional receiver-driven window-based methods, the Interest Control Protocol (ICP) [6] and the Content Centric TCP (CCTCP) [7] follow the traditional TCP window control, 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. ICP uses one timer for one flow, just like the TCP round-trip time (RTT) estimation mechanism. CCTCP introduces a timer for an individual Interest packet by inserting a timestamp in Interest and Data packets. The Chunk-switched Hop Pull Control Protocol (CHoPCoP) [8] is another window-based method that introduces the explicit congestion notification with random early marking and changes the window size according to the AIMD mechanism with constant value increasing in the additive increase. CHoPCoP introduces an Interest packet shaper at an intermediate router.

Recently proposed window-based methods adopt new approaches. The CUBIC-based method [9] follows the CUBIC TCP congestion control in its window control, with an explicit congestion tag in a Data packet. The Practical Congestion Control (PCON) [11] and the Window based Congestion Control Mechanism (WinCM) [12] introduces an active queue management monitoring packet-sojourn time at routers. PCON can implement a number of classic

Table I. Related work on receiver-driven window-based approach

Method	Congestion handling	Window control	Comments
ICP [6]	timeout	AIMD (only in congestion avoidance phase, similar with TCP Reno)	one timer per flow
CCTCP [7]	timeout	AIMD (slow start and congestion avoidance phases, similar with TCP Reno with timestamp option)	one timer per Interest, timestamp in Interest
CHoPCoP [8]	congestion tag, timeout	AIMD (slow start and congestion avoidance phases, increase by constant value in congestion avoidance phase)	Interest shaping at routers
CUBIC-based [9]	congestion tag, timeout	CUBIC TCP	queue monitoring at routers
PCON [11]	congestion tag, NACK, timeout	TCP Reno or BIC TCP, one decrease per RTT	CoDel at routers, consider multipath
WinCM [12]	congestion tag, NACK, timeout	no congestion: BIC TCP, light congestion: TCP Reno, additive decrease	AQM at routers, consider multipath
VCP-based [13]	load factor	multiple increase mechanism dependent on load factor, multiplicative decrease	-

TABLE II. RELATED WORK ON RATE-BASED APPROACH

Method	Congestion handling	Rate control	Comments
SF [14]	NACK, timeout	heuristic, eg. AIMD	-
new SF [15]	NACK, timeout	heuristic, eg. AIMD, one decrease per RTT	-
HoBHIS [16], [17]	suppressed by rate reporting	follow reported rate	calculate rate based on Interest rate to upstream, downstream link bandwidth, # of queued Data packets, RTT with producer, # of flows
ECN [18]	suppressed by rate reporting	follow reported rate	focusing on upstream link parameters, does not use # of flows
MIRCC [19]	suppressed by rate reporting	follow reported rate	strictly focusing on upstream link, consider multipath
MNRCP [20]	suppressed by rate reporting	follow reported rate	use # of Interest and Data flows, use NACK, consider multipath

loss-based TCP algorithms, and actually, implemented TCP Reno and BIC TCP. PCON avoids unnecessary window reduction resulting from consecutive negative acknowledgments (NACKs) or congestion tags, and allows at most one window decrease within one RTT. WinCM introduces Reno and BIC TCP. During no congestion, i.e. when the cumulative count of congestion tagged Data packets is zero, the window increase follows BIC TCP, and otherwise it follows TCP Reno. As for the window decrease, it adopts the additive decrease instead of the multiplicative decrease. The Variable-Structure Congestion Control Protocol (VCP)-based method [13] introduces the load factor that indicates the status of Data packet sending queue in an individual router. The largest load factor in a flow is conveyed in a Data packet. A consumer adjusts the window increasing according to the load factor for a flow. In the case of low load, the multiplicative increase is used. When moderate load and high load, the additive increase with constant value and the Reno-like additive increase are used, respectively. The window decreasing adopts the multiplicative decrease.

It is considered that those window-based methods have a problem that the window size itself may not be optimal when Data packets are cached in different routers.

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 reporting scheme, in which intermediate routers report the optimal Interest rate to a consumer. The Stateful Forwarding (SF) [14] is an example of the former scheme. It introduces an NACK packet, which has a similar packet structure with Interest, as a negative response to an Interest packet. NACK packets are generated when a router losses an Interest packet, e.g., due to the congestion detection. The new SF [15] is an extension of SF in order to avoid unnecessary rate reduction due to multiple NACKs generated during one congestion event. It reduces the rate once within one RTT, as PCON mentioned above.

In contrast with those non-deterministic methods, there are some methods that enable routers to report a maximum allowed Interest sending rate to a consumer [16]-[20]. These methods take a similar approach but have several differences in the detailed procedure. The Hop-By-Hop Interest Shaping (HoBHIS) [16], [17] is one proposed in an early stage. A router focuses on the upstream link for the Interest packet sending and on the downstream link for the Data packet sending. It also use the number of flows, whose monitoring provides significant overheads for routers. The Explicit Congestion Notification (ECN) based Interest sending rate control method [18] tries to focus only on the upstream link, but it seems to still use the Data packet queue length on the downstream link. MIRCC [19] is sophisticated compared with other methods, in the meaning that it just focuses only on the upstream link. The Interest packet queue length is used instead of the Data packet queue length in other methods. The Rate-based, Multipath-aware Congestion Control Algorithm (MNRCP) [20] is based on a similar scheme with HoBHIS, and it takes account of the numbers of Interest and Data flows separetely.

These rate-based methods with explicit rate reporting are able to control Interest transmission so as to suppress congestion, and as a result they can provide higher throughput. However, their implementation requires the precise timing control for sending Interest packets. The fine-grained clock is hard to implement in off-the-shelf computers, as discussed later. The Hop-by-hop Window-based Congestion Control (HWCC) [24] tries to resolve this problem by taking the hop-by-hop window-based approach. HWCC introduces a hop-by-hop acknowledgment (H-ACK) packet that notifies a router of the reception of an Interest packet together with the Interest rate over the next link to the producer. The per-hop window size between this router and the next router is determined according to the reported rate and the link RTT measured by the H-ACK.

B. Overhead of fine-grained clock implementation

In off-the-shelf computers, the rate control mechanism in which an Interest packet is sent in a specific interval is implemented by the interrupt handling framework. As described in the previous section, it is required to handle various Interest sending rates, and so the interrupting clock tick takes finer value. In an example given above, the clock tick is 1 μsec while the actual sending intervals are 10 μsec, 20 μsec, or 33 μsec.

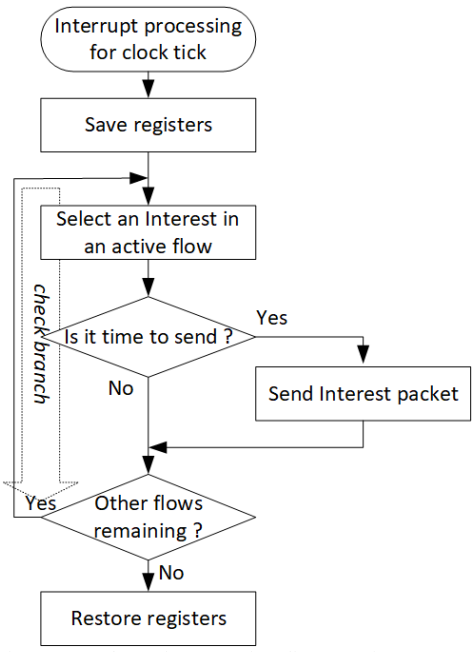


Figure 1. Schematic processing diagram of rate control

Figure 1 shows a schematic processing diagram of the rate control in NDN nodes. A timer hardware generates clock tick interrupts regularly, and then the interrupt handling shown in Figure 1 is invoked. At the beginning, the current values of registers are saved. Then, an Interest packet of an Interest-Data flow stored in a send queue is selected, and it is checked whether the time to send the Interest packet is reached or not. If the sending time is not reached, then an Interest packet of another flow is searched and the same process is done in the case there is another one, otherwise the interrupt handling routine is finished after restoring the register values.

Here, we assume that the CPU clock rate is 4GHz, and that the processing flow when the sending time is not reached (No for "Is it time to send ?" in the figure, *check branch*) requires forty CPU clocks. Then, the time for one check branch is 10 nano sec (1/4 nano sec × 40). If an NDN node handles 100 active flows, the interrupt handling routine takes 1 μsec to be executed even if there are no Interest packets to be sent at that timing. If the clock tick for the rate control is 1 μsec, one CPU executes only the check branches by its full capability. Recent CPUs have multiple cores such as 8 cores. If the number of active Interest-Data flows is 1,000, recent multi-core CPUs cannot support even the check branches.

There are some traditional rate-based schemes, but they use some hardware mechanism instead of the fine-grained clock. For example, the Asynchronous Transfer Mode (ATM) uses a kind of rate-based cell transfer [26], but ATM uses null cells discarded at a receiving side to regulate cell streams at a specific rate. [27] introduced pause packets over Gigabit Ethernet, corresponding to null cells in ATM, that are used only between end nodes and switching hubs. Those approaches are implemented by the MAC level hardware, but NDN Interest packets are handled as a higher level, which makes the hardware support difficult.

TABLE III.
LIST OF MIRCC PARAMETERS

Parameter	Definition
$R(t)$	Stamping rate at time t
C	Capacity of upstream link
N	Equivalent number of flows with full rate
T	Interval of rate calculation
$q(t)$	Inflated instantaneous queue size
$y(t)$	Incoming Interest rate during $[t-T, t)$
$d(t)$	Smoothed average RTT
$\beta(t)$	Self-tuned parameter for stability
η	Target link utilization

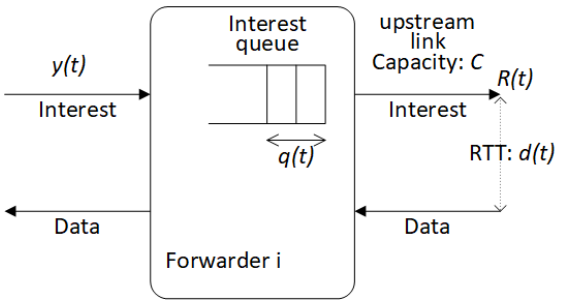


Figure 2. Forwarder model in MIRCC

C. Details of MIRCC

In order to examine the impact of coarse-grained clock to the rate reporting congestion control, we pick up MIRCC. In MIRCC, consumers and routers that forward Interest packets, called forwarders, maintain the parameters indicated in Table 3. The model of a forwarder is given in Figure 2, which explains some of the parameters. It should be noted that the parameters focus on the upstream link of a forwarder. Each forwarder calculates the Interest sending rate $R(t)$ for individual flows, at each interval T. $R(t)$ is specified as the sum of *base_rate(t)* and *excess_rate(t)*. The *base_rate(t)* is the rate to split the allowed link bandwidth among the passing flows. The *excess_rate(t)* is for filling the extra available bandwidth with traffic equally. Each of them is given in the following way.

In order to calculate *base_rate(t)*, a forwarder estimates the number of flows by equation (1).

$$N = {max(C, y(t))}/{R(t-T)} \quad (1)$$

Then, *base_rate(t)* is computed as follows:

$$base\_rate(t) = \frac{\eta C - \beta(t)\frac{q(t)}{d(t)}}{N} \quad (2)$$

Here, $\beta(t)$ is given by

$$\beta(t) = max\left(0.1, \frac{y(t)-y(t-T)}{y(t)}\right) \quad (3)$$

As for *excess_rate(t)*, the following equation is used.

$$excess\_rate(t) = R(t-T) - {}^{y(t)}/_{N} \quad (4)$$

In order to avoid high-frequency oscillation, an exponential weighted moving average (EWMA) is applied to both *base_rate(t)* and *excess_rate(t)* with weight 0.5. Finally, *R(t)* is given by the following equation.

$$R(t) = base\_rate\_ewma(t) + excess\_rate\_ewma(t) \quad (5)$$

When a router receives a Data packet, it checks the stamping rate included in the packet, and if the included rate is larger than the computed *R(t)*, then *R(t)* is set in the Data packet.

III. MIRCC WITH COARSE-GRAINED CLOCK

A. Implementation of MIRCC over ndnSIM

In order to evaluate the performance of MIRCC, we implemented it over the ndnSIM simulator version 1.0 [28]. The reason we used this version is that we reused the coarse-gained clock implementation in our previous paper. We implemented MIRCC in the following way.

(1) Add R(t) parameter in Data packet

In order to convey *R(t)* in a Data packet, we defined the corresponding parameter, `m_rate`, and the methods to access and modify it, in files `model/ndn-data.{h,cc}`. Besides, the methods for formatting a Data packet, `Serialize()` and `Deserialize()`, is modified in `model/wire/ndnsim.cc`.

(2) Implement a method calculating R(t)

A method called `CalculateRate()` is implemented in `utils/ndn-limits.cc`. This method is invoked every *T* interval, and calculates *R(t)* according to equations (1) through (5) specified above. Here, we need to mention that *y(t)* is given by dividing the number of received Interest packets by *T*, that the leaky bucket size is used as *g(t)*, and that η=1 in our case.

(3) Add various functions in Interest/Data handling

Interest and Data packets are handled in file `model/fw/ndn-forwarding-strategy.cc`. We added the following functions in this file.

- Counting received Interest packets.
- Evaluating smoothed RTT at receiving Data packets.
- Setting *R(t)* in a Data packet if the corresponding value in the Data packet is larger than the calculated *R(t)*.

(4) Behaviour of consumer

A consumer sends Interest packets according to the reported *R(t)* in Data packets. We implemented this kind of consumer as a new class called `ConsumerLi`.

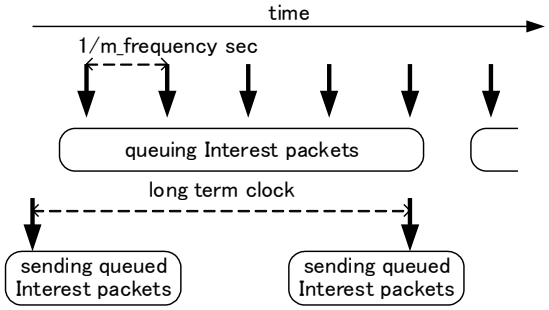


Figure 3. Implementation scheme of coarse-grained clock system in a consumer

B. Implementation of Coarse-grained Clock Based MIRCC

In the original NDN, the rate control in the consumer is implemented as follows. The sending of Interest packets with a specific rate is done in the `ScheduleNextPacket()` method of the `ConsumerLi` class. In this method, the `SendPacket()` method of the `Consumer` class, which is the superclass, is invoked periodically, every `1.0/m_frequency` seconds. The `SendPacket()` method sends one Interest packet actually.

We emulated a course-grained clock in the `Consumer` class in the following way (Figure 3).

- A clock system with longer tick, such as 100 msec, is implemented in the `ConsumerLi` 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.

In the router side in MIRCC, we implemented a coarse-grained clock, by assigning a large value, such as 100 msec, in the interval *T*.

IV. PERFORMANCE EVALUATION MIRCC WITH COARSE-GRAINED CLOCK

A. Experimental setup

The network configuration used in this evaluation is shown in Figure 4, which is a dumbbell configuration where two consumers (C1 and C2) and two producers (P1 and P2) are connected through two routers (R1 and R2). The bandwidth and delay between a consumer and a router, and between a router and a producer are 10Mbps and 50 msec, respectively. Those between routers are 12Mbps and 100 msec, respectively. The length of a Data packet is 1,250 bytes (10,000 bits), and so the link speed 10Mbps and 12Mbps corresponds 1,000 packets/sec and 1,200 packets/sec, respectively. The depth of a token bucket for

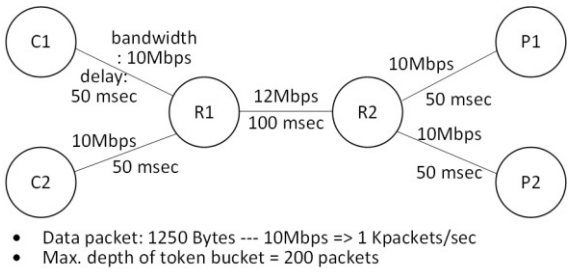


Figure 4. Network configuration for performance evaluation

policing the Interest packets is set to 200 packets in consumers and routers.

We assume that all of the nodes in Figure 4 use the same granularity for the rate control clock. That is, the long-term clock in the consumers and interval T in the routes uses the same time interval value. Under these conditions, we evaluated the cases that the rate control clock has 100 msec, 250 msec, 500 msec and 750 msec interval values. In all the evaluation runs, consumer C1 transmits Interest packets to producer P1 between time 1 sec and 10 sec, and consumer C2 sends Interest packets to producer P2 between time 3 sec and 8 sec. In this evaluation, cache is not used.

B. Results of performance evaluation

Figure 5 shows the results, where the time interval is set to 100 msec, 250 msec, 500 msec, and 750 msec. The graphs show the Interest sending rate at consumers C1 and C2.

The result in Figure 5(a) corresponds to the case that the time interval is 100 msec. In the beginning, the Interest sending rate at C1 takes the value around 1,000 packets/sec although there is a slight fluctuation. When C2 starts the communication, the Interest sending rates at C1 and C2 go to 600 packets/sec, with some fluctuations. This result shows that two flows from C1 and C2 share the bandwidth of the bottleneck link evenly. It is considered that MIRCC performs well, even if the time interval is relatively large.

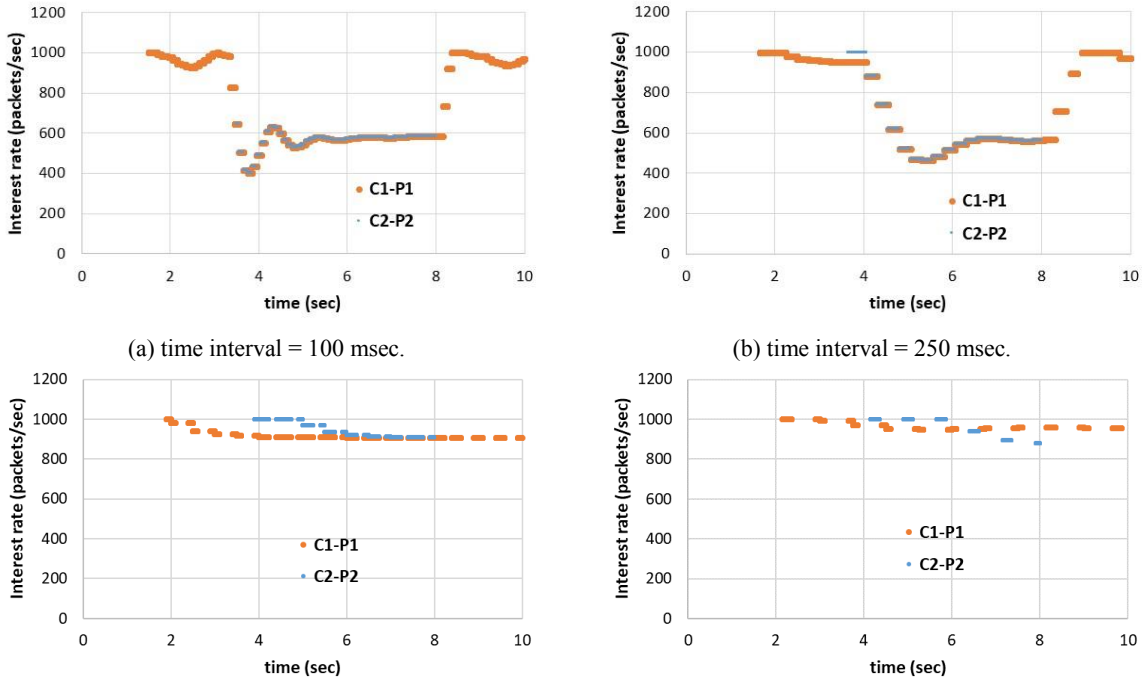
Figure 5(b) shows the result in the case that the time interval is 250 msec. The result seems to be similar with the case of 100 msec. The difference is that the convergence to 600 packets/sec when C2 starts the communication takes larger period, around 2 sec.

In contrast to these results, those in Figures 5(c) and 5(d) show a catastrophic situation. Even if C2 starts its session, the Interest sending rates in C1 and C2 keep 1,000 packets/sec as if the individual consumer communicates alone. These results indicate that the MIRCC mechanism does not work well. It should be noted that the time interval values in those cases are larger than RTT (400 msec) between the consumer and the producer.

We also examined the Data packet delivery losses. Table 4 shows the loss rate of the C1-P1 and C2-P2 flows in the four coarse-grained clock values. These values give the overall loss rate throughout individual flows. It should be

TABLE IIV. OVERALL LOSS RATE OF EACH FLOW

time interval / flow	100 msec	250 msec	500 msec	750 msec
C1-P1	0.017	0.103	0.552	0.704
C2-P2	0.036	0.239	0.576	0.712



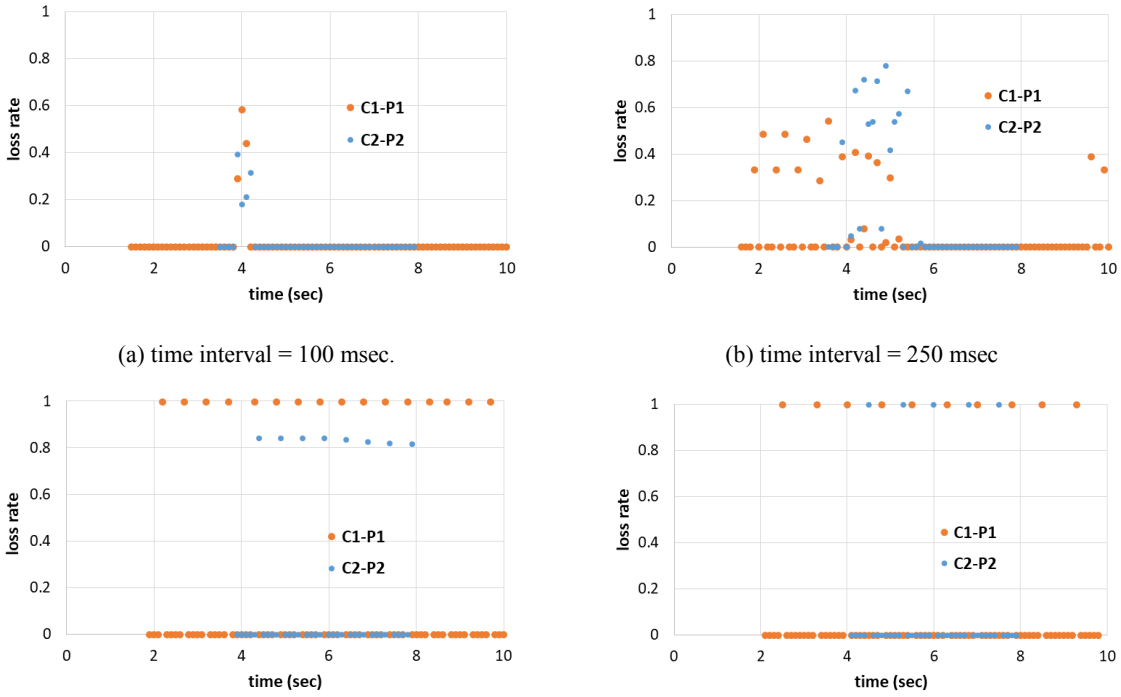
(a) time interval = 100 msec.

(b) time interval = 250 msec.

(c) time interval = 500 msec.

(d) time interval = 750 msec.

Figure 5: Time variation of Interest sending rate.



(a) time interval = 100 msec.

(b) time interval = 250 msec

(c) time interval = 500 msec.

(d) time interval = 750 msec.

Figure 6. Time variation of Data packet loss rate.

noted that our performance evaluation suppresses the retransmissions of Interest packets. So, the loss rate given here is that for original Interest packets. That is, the loss rate is given by the number of delivered Data packets divided by the number of sent Interest packets. In the case that the time interval is 100 msec, the loss rate is small, and that for 250 msec time interval is 0.1 for the C1-P1 flow and 0.24 for the C2-P2 flow. On the other hand, the loss rates for 500 msec and 750 msec time intervals is larger than 0.5. These correspond to the result of the Interest sending rate vs. time that the MIRCC congestion control does not work well under these time interval values.

Figure 6 shows the time variation of the Data packet loss rate calculated for every 100 msec. Figure 6(a) shows that there are some Data packet losses around 4 sec, both in the C1-P1 and C2-P2 flows. This time frame is just after the C2-P2 flow started. In other time frames, all Data packets are delivered. As a result, the overall loss rate is low in the case of 100 msec time interval.

When the time interval is 250 msec, there are some losses from time 2 sec to 4 sec (Figure 6(b)). This means that there are some losses in the C1-P1 flow while only this flow exists. When the second flow started, there are some losses between time 4 sec and 5.5 sec. These losses happen while the Interest sending rate of two flows decreases from 1,000 packets/sec to 600 packets/sec. The reason for these losses is that this decreasing is slower than the case of 100 msec time interval. This delay is considered to be caused by the slow behavior of consumers and routers due to longer time interval.

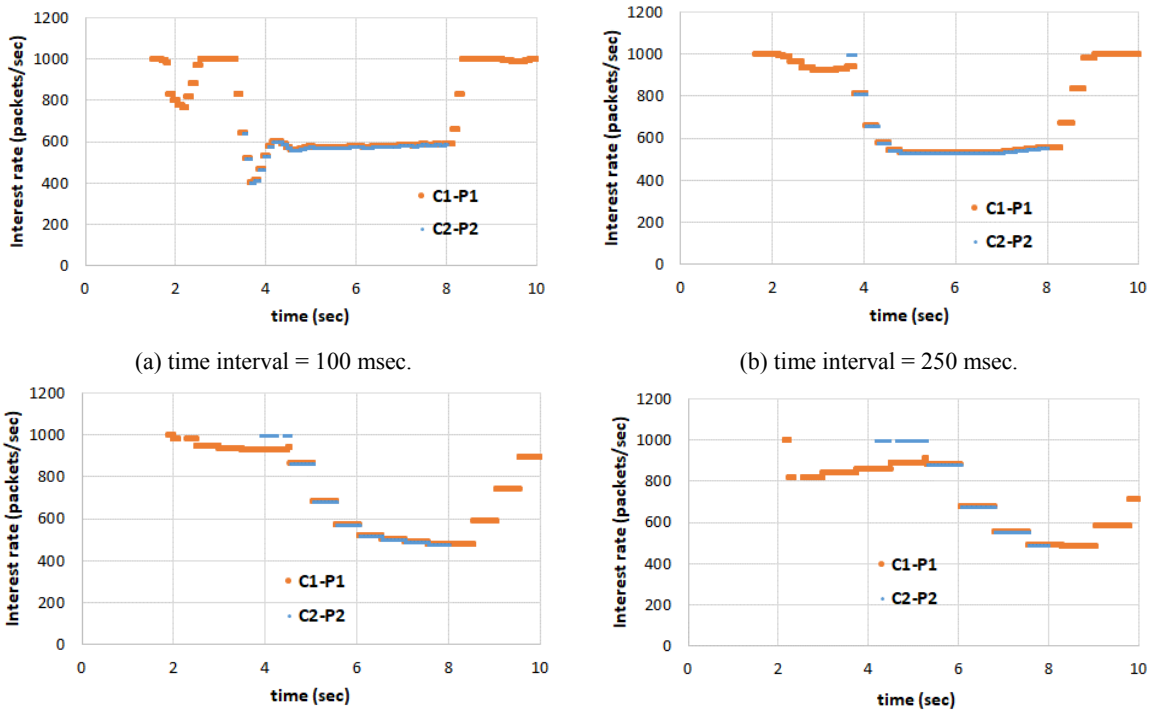
The results in Figures 6(c) and 6(d) are significantly deferent. While the C1-P1 flow is continuing by itself, there are some periods when the loss rate is 100%. This means that an MIRCC flow does not work well in the situation that the shaping clock tick in a consumer and interval T in routers have a large value. As a result, the loss rate becomes larger than 50%, and so the mechanism cannot detect the fact that there are two flows sharing the bottleneck link.

V. Smoothening Interest Sending and Its Performance Evaluation

A. Smoothening Interest sending

In this section, we propose a mechanism that smoothens the Interest packet sending without using fine-grained clock. The performance degradation shown in the previous section comes from long time intervals used in consumers and routers. The long time interval in consumers makes the Interest packet sending bursty, and that in routers makes the update of the stamping rate unfrequent. Although the rate update in routers is difficult to make frequent, the burstiness of Interest sending at consumers can be reduced by a mechanism to process the Interest sending when Data and NACK packets are received. This mechanism was useful for the Stateful Forwarding with coarse-grained clock [22].

Here, we propose an Interest control method that utilizes the Data and NACK packet receiving timing. The receiving processing of Data and NACK packets is triggered by a packet receive interrupt. This does not require large



(a) time interval = 100 msec.

(b) time interval = 250 msec.

(c) time interval = 500 msec.

(d) time interval = 750 msec.

Figure 7. Time variation of Interest sending rate with Interest sending smoothening.

processing overhead, different from the software based rate control mechanism. So, the receiving timing is a good chance to proceed the Interest packet sending. We have added the following mechanism in the coarse-grained clock based MIRCC described in Subsection III.B.

- When a consumer receives a Data or a NACK packet, it processes the received packet and then tries to send the Interest packets that need to be sent by this timing, i.e., those stored in the Interest queue in the implementation described in Subsection III.B.
- This procedure is implemented in the `OnData()` and `OnNack()` methods in the `Consumer` class.

B. Performance evaluation of coarse-grained clock based MIRCC with Interest sending smoothening

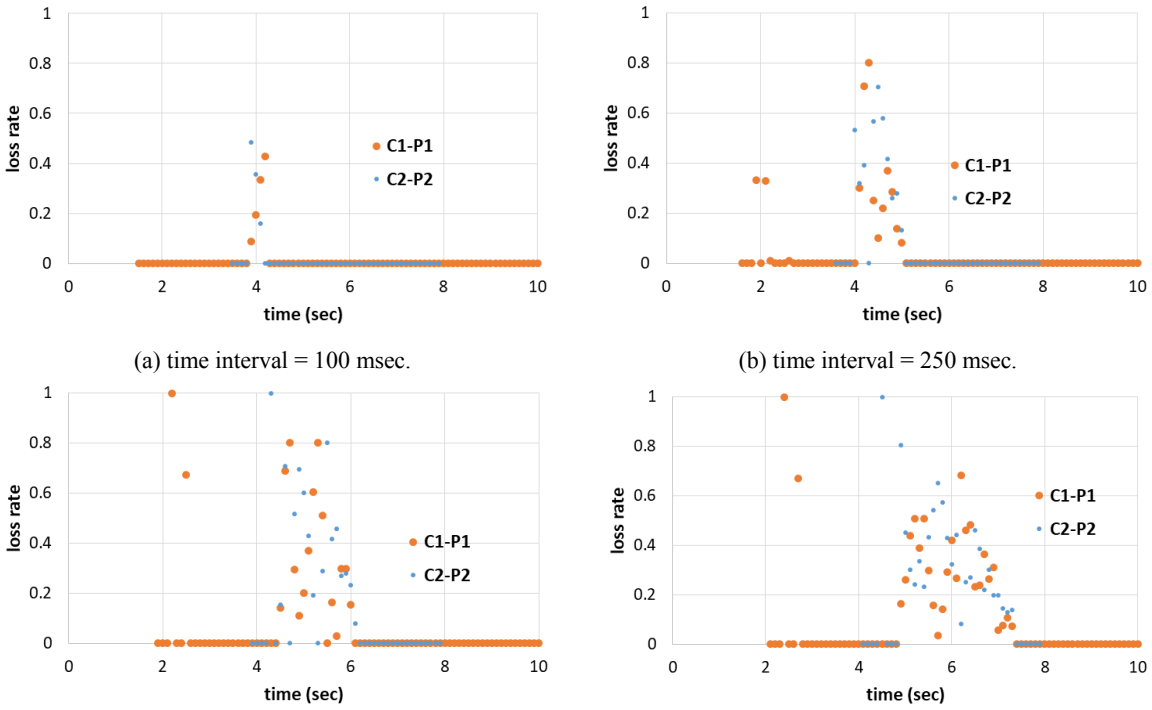
This subsection shows the result of performance evaluation of coarse-grained clock based MIRCC with Interest sending smoothening. We use the same experimental conditions described in Subsection IV.A.

Figure 7 shows the Interest packet sending rate of the proposed method, when the time interval value is 100 msec, 250 msec, 500 msec, and 750 msec. The results in Figures 7(a) and 7(b) are similar to those shown in Figures 5(a) and 5(b), respectively. This means that, when the original MIRCC is working well, the Interest sending smoothening proposed here does not contribute so much to the performance.

The results shown in Figures 7(c) and 7(d) differ largely from those in Figures 5(c) and 5(d), respectively. When the time interval used in consumers and routers is large, the original MIRCC cannot estimate the optimal rate. As shown in Figure 6(c) and 6(d), there are packet losses even when only one flow exists. Since the data links used by the C1-D1 flow have enough bandwidth when only this flow exists, these packet losses seem to come from the Interest packet shaping using the coarse-grained timer at the consumer. The Interest sending smoothening proposed here, on the other hand, distributes the Interest packet sending to the timing when Data or NACK packets are received. As a result, the Interest sending rates of two flows could be tuned to the half of the bottleneck link bandwidth. That is, the proposed smoothening method takes an effect similar to the Interest packet sending with smaller time intervals, although the Interest sending itself is not performed regularly in an identical interval. Comparing the results in Figure 7(c) and 7(d), the shift of Interest sending rate from 1,000 packets/sec to 600 packets/sec is slower in the case of 750 msec interval. This is because the new stamping rate is reported by routers at the configured time interval, and the consumers take longer time to change the rate in the case of 750 msec interval.

TABLE V. OVERALL LOSS RATE OF EACH FLOW WITH INTEREST SENDING SMOOTHENING

time interval / flow	100 msec	250 msec	500 msec	750 msec
C1-P1	0.009	0.062	0.156	0.209
C2-P2	0.034	0.145	0.308	0.399



(a) time interval = 100 msec.

(b) time interval = 250 msec.

(c) time interval = 500 msec.

(d) time interval = 750 msec.

Figure 8. Time variation of Data packet loss rate with Interest sending smoothening.

Table 5 and Figure 8 shows the results of Data packet loss rate. Table 5 shows the overall loss rate of individual flows calculated throughout the sessions. Comparing with the results in Table 4, the loss rates when the time interval is 100 msec and 250 msec are similar for the original MIRCC and the MIRCC with Interest sending smoothening. More specifically, the loss rate of the smoothing is slightly smaller than the original MIRCC. For the cases of 500 msec and 750 msec time interval values, the overall loss rate in the smoothening is much better than the original MIRCC, whose loss rate was larger than 50%.

Figure 8 shows the time variation of the Data packet loss late calculated for every 100 msec. By comparing Figures 6(a) and 8(a), the smoothening reduced the Data packet loss rate slightly in the case of 100 msec time interval. Figure 8(b) shows some difference from Figure 6(b) in the case of 250 msec time interval. By introducing the smoothening, the Data packet losses are limited to the time frames of around 2 sec and from 4 sec to 5 sec.

The results in Figures 8(c) and 8(d) changed largely from those in Figures 6(c) and 6(d). By introducing the smoothening, the Data packet losses in the single flow are reduced largely in the cases of 500 msec and 750 msec time intervals. When two flows exist in the case of 500 msec and 750 msec time intervals, the smoothening also reduced the loss rate. While there were several 100 % loss rate observations in the original MIRCC, there is just one 100 % loss rate observation by use of the smoothening.

VI. Conclusion

This paper discussed about the impact of the coarse-grained clock on MIRCC, one of rate-based NDN congestion control methods with explicit rate reporting. We implemented MIRCC over the ndnSIM simulator and evaluated the performance in a dumbbell network when the rate control clocks in consumers and routers have a large tick value. The result showed that the rate control of MIRCC does not work well due to bursty Interest packet sending at consumers and rough rate detection at routers. We proposed a method which smoothens the Interest packet sending by use of the timing of Data packet receptions. The results of performance evaluation showed that the smoothening is effective in reducing the Data packet losses when the rate control clock has a large tick value.

References

[1] Cisco public, *Cisco Annual Internet Report (2018-2023)*. White paper, 2020.

[2] V. Jacobson et al., "Networking Named Content," in *Proc. CoNEXT '09*, pp. 1-12.

[3] 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.

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

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

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

[7] L. Saino, C. Cocora, and G. Pavlou, "CCTCP: A Scalable Receiver-driven Congestion Control Protocol for Content Centric Networking," in *Proc. IEEE ICC 2013*, pp. 3775-3780.
[8] F. Zhang, Y. Zhang, A. Reznik, H. Liu, C. Qian, and C. Xu, "A Transport Protocol for Content-Centric Networking with Explicit Congestion Control," in *Proc. IEEE ICCCN 2014*, pp. 1-8.
[9] Y. Liu, X. Piao, C. Hou, and K. Lei, "A CUBIC-Based Explicit Congestion Control Mechanism in Named Data Networking," in *Proc. IEEE CyberC 2016*, pp. 360-363.
[10] K. Nichols and V. Jacobson, "Controlling Queue Delay," *ACM Magazine Queue*, vol. 10, issue 5, pp. 1-15, May 2012.
[11] K. Schneider, C. Yi, B. Zhang, and L. Zhang, "A Practical Congestion Control Scheme for Named Data Networking," in *Proc. ACM ICN 2016*, pp. 21-30.
[12] M. Wang, M. Yue, and Z. Wu, "WinCM: A Window based Congestion Control Mechanism for NDN," in *Proc. IEEE HotICN 2018*, pp. 80-86.
[13] S. Xing, B. Yin, J. Yao, H. Zhang, Q. Zhai, and H. Shi, "A VCP-based Congestion Control Algorithm in Named Data Networking," in *Proc. IEEE IAEAC 2018*, pp. 463-468.
[14] 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.
[15] T. Kato and M. Bandai, "Congestion Control Avoiding Excessive Rate Reduction in Named Data Network," in *Proc. IEEE CCNC 2017*, pp. 1-6.
[16] N. Rozhnova and S. Fdida, "An effective hop-by-hop Interest shaping mechanism for CCN communications," in *Proc. IEEE INFOCOM Workshops 2012*, pp. 322-327.
[17] N. Rozhnova and S. Fdida, "An extended Hop-by-hop Interest shaping mechanism for Content-Centric Networking," in *Proc. IEEE GLOBECOM 2014*, pp. 1198-1204.
[18] J. Zhang, Q. Wu, Z. Li, M. A. Kaafar, and G. Xie, "A Proactive Transport Mechanism with Explicit Congestion Notification for NDN," in *Proc. IEEE ICC 2015*, pp. 5242-5247.
[19] M. Mahdian, S. Arianfar, J. Gibson, and D. Oran, "Multipath-aware ICN Rate-based Congestion Control," in *Proc. ACM ICN 2016*, pp. 1-10.
[20] S. Zhong, Y. Liu, J. Li, and K. Lei, "A Rate-based Multipath-aware Congestion Control Mechanism in Named Data Networking," in *Proc. IEEE ISPA/IUCC 2017*, 174-181.
[21] K. Fall and W. Stevens, *TCP/IP Illustrated, Volume1; The Protocols, Second Edition*. Addison-Wesley, 1994.
[22] T. Kato, K. Osada, R. Yamamoto, and S. Ohzahata, "A Study on How Coarse-grained Clock System Influences NDN Rate-based Congestion Control," in *Proc. IARIA ICN 2018*, pp. 35-40.
[23] T. Kato, T. Enda, R. Yamamoto, and S. Ohzahata, "A Study on Performance of Explicit Rate Report Based Congestion Control under Coarse-grained Clock Management," in Proc. *INSTICC DCNET 2020*, pp. 82-88.
[24] T. Kato and M. Bandai, "A Congestion Control Method for NDN Using Hop-by-hop Window Management," in *Proc. IEEE CCNC 2018*, pp. 1-6.
[25] A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM: NDN simulator for NS-3," *NDN, Technical Report NDN-0005*, 2012.
[26] ITU-T, *B-ISDN asynchronous transfer mode functional characteristics, Series I: Integrated Services Digital Network*. Recommendation I.150, Feb. 1999.
[27] Y. Yamamoto, "Estimation of the advanced TCP/IP algorithms for long distance collaboration," *Fusion Engineering and Design*, vol. 83, issue 2-3, pp. 516-519, Apr. 2008.
[28] NDN, "Overall ndnSIM documentation; Forwarding Strategies," http://ndnsim.net/1.0/fw.html.