Journal of Information Processing Vol.20 No.1 185–195 (Jan. 2012)

[DOI: 10.2197/ipsjjip.vol.20.185]

Regular Paper

DP-FEC: Dynamic Probing FEC for High-Performance Real-Time Interactive Video Streaming

Kazuhisa Matsuzono[1,a)] Hitoshi Asaeda[1] Jun Murai[1]

Received: April 11, 2011, Accepted: September 12, 2011

Abstract: High-quality and high-performance real-time interactive video streaming requires both keeping the highest data transmission rate and minimizing data packet loss to achieve the best possible streaming quality. TCP-friendly rate control (TFRC) is the most widely recognized mechanism for achieving relatively smooth data transmission while competing fairly with TCP flows. However, because its data transmission rate depends largely on packet loss conditions, high-quality real-time streaming suffers from a significant degradation of streaming quality due to both a reduction in the data transmission rate and data packet losses. This paper proposes the dynamic probing forward error correction (DP-FEC) mechanism that is effective for high-quality real-time streaming to maximize the streaming quality in a situation in which competing TCP flows pose packet losses to the streaming flow. DP-FEC estimates the network condition by dynamically adjusting the degree of FEC redundancy while trying to recover lost data packets. It effectively utilizes network resources and adjusts the degree of FEC redundancy to improve the playback quality at the user side while minimizing the performance impact of competing TCP flows. We describe the DP-FEC algorithm and evaluate its effectiveness using an NS-2 simulator. The results show that by effectively utilizing network resources, DP-FEC enables to retain higher streaming quality while minimizing the adverse condition on TCP performance, thus achieving TCP friendliness.

Keywords: real-time streaming, forward error correction, dynamic FEC, TCP-friendly rate control

1. Introduction

The evolution of the Internet has seen a rapid growth in both users and applications. Owing to the dissemination of high-speed broadband networks, users are expecting to fulfill an application demand more comfortably, and applications promote users to increase their motivation. High-quality and high-performance real-time applications are the primary ones. Engineers are developing high-performance streaming applications [5], [23] that can be used easily with consumer electronic devices and ordinary PCs for the purposes of e-learning, international symposiums, and telemedicine [2]. It is easily expected that as the Internet evolves to higher-speed networks, these video applications will become increasingly popular.

Typically, real-time interactive video streaming requires the minimization of packet losses while maintaining the highest data transmission rate under acceptable transmission delays, to produce the best possible streaming quality. However, because high-quality and high-performance streaming flows consume a large amount of network bandwidth for data transmission, they may compete for bandwidth with other data flows (e.g., transmission control protocol (TCP) and/or user datagram protocol (UDP) flows) in a shared best-effort network like the Internet, and cause packet losses that can severely impact playback quality. The quality degradation caused by packet loss is a critical problem, especially for mission-critical applications such as telemedicine. The type of packet loss caused by congested routers, which is characterized by near-random and unpredictable behavior, is inevitable even with the future Internet, and therefore presents a formidable challenge to high-performance streaming applications seeking to optimize streaming quality.

TCP-friendly rate control (TFRC) has become the de facto standard to control network congestion [10], [12]. In accordance with the definition of TCP friendliness, TFRC tries to maintain fairness with competing TCP flows in the same network condition while providing a promising mechanism for a smooth data transmission rate [17], [27]. The throughput of non-TCP flows is strongly affected by the average throughput of a conformant TCP connection under the same conditions. Since this constraint can force a high-performance streaming flow to reduce the data transmission rate at the expense of video quality, TFRC is not suitable for the flow to maximize the streaming quality. In addition, when the feedback delay (i.e., the round trip time) is large, a timely and correct estimation of network conditions becomes an impossible task. As a result, unacceptable conditions for video streaming (i.e., packet loss or data rate oscillations) continuously occur [24], and they can have a negative impact on the communication quality of other competing flows (e.g., throughput). In this context, existing approaches do not properly control congestion to fulfill the demands of end-users wanting the highest streaming quality.

To address the problem that TFRC fails to maintain higher streaming quality, we propose the dynamic probing forward er-

1 Graduate School of Media and Governance, Keio University, Fujisawa, Kanagawa 252–0882, Japan

a) kazuhisa@sfc.wide.ad.jp

ror correction (DP-FEC) mechanism that provides a promising method for achieving higher streaming quality while seeking the behaviors of competing TCP flows. DP-FEC estimates the network conditions by dynamically adjusting the degree of FEC redundancy while trying to recover lost data packets. By successively observing variation in the intervals between packet loss events, DP-FEC effectively utilizes network resources. It optimally adjusts the degree of FEC redundancy to recover as many lost data packets as possible while minimizing the performance impact of competing TCP flows. We evaluated the DP-FEC algorithm using an NS-2 simulator, and showed that it allows streaming flows to retain higher streaming quality and minimizes the impact of FEC on TCP performance to achieve TCP friendliness.

The remainder of this paper is organized as follows: Section 2 examines the requirements for high-performance real-time video streaming. Section 3 gives an overview of the DP-FEC algorithm. Section 4 evaluates the effectiveness of DP-FEC using an NS-2 simulation. Section 5 describes related work, and in Section 6, we present our conclusions.

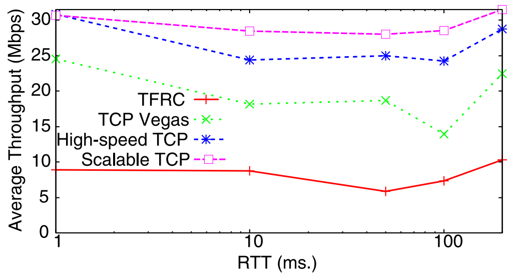
2. High-Quality Real-Time Video Streaming over IP

2.1 Background and Problem Statements

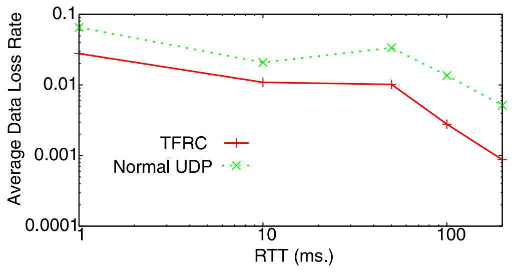
Most of the proposed congestion control mechanisms for real-time video streaming applications try to achieve TCP-friendliness according to its definition [1], [10], [11], [17], [27]; TFRC thus enables a streaming flow to maintain a desired smoothness of data transmission rate and fairness with coexisting TCP flows. However, because high-performance video streaming requires a large amount of bandwidth that cannot be utilized by TCP, throughput regulation based on TCP-friendliness reduces the data transmission rate at the expense of a reduction in video quality (i.e., the transmission rate is adjusted by changing the parameters for encoding the video). Video quality is increasingly reduced as the round trip time (RTT) increases, because TCP is inefficient under network conditions in which the RTT is large [*1]. In addition, because TFRC examines packet loss conditions as an indicator of congestion, it often delays the action for rate control and causes the degradation of playback quality. Estimated RTT variations are also utilized as a congestion indicator to avoid packet losses caused by traffic congestion. Yet, delay-based congestion control mechanisms [13], [14] (e.g., TCP Vegas) react to RTT variations incorrectly and cause a severe quality degradation, especially in high bandwidth paths [15], [16].

To clarify the specific problem of competition between a high-performance real-time streaming flow and TCP flows, we used an NS-2 simulator to simulate a single bottleneck with a 100 Mbps link capacity. The round-trip propagation delay varied from 1 to 200 ms. DropTail queue is used and the queue size is set to max{100, Bandwidth Delay Product (BDP)}. To create network congestion, we generate short-lived TCP flows using a Poisson process with an average rate of 30 flows per second, and

[*1] A high-performance TCP protocol (such as scalable TCP and high-speed TCP) could be used in real-time streaming, but it typically results in some quality degradation (such as a longer startup delay and lack of smoothness).



(a) Average throughput of a single flow (the cases of TFRC/TCP-Vegas/HSTCP/Scalable-TCP)



(b) Average data loss rate of a single flow (the cases of TFRC/normal-UDP)

Fig. 1 Average throughput of a single streaming flow, and the average data loss rate of a single streaming flow under varied RTTs.

the size of a TCP flow follows Pareto distribution with an average of 180 packets and a shape parameter of 1.5. As a high-performance real-time streaming, we used 1) a TFRC flow, 2) a TCP-Vegas [13] flow, 3) a High-speed TCP (HSTCP) [19] flow, 4) a Scalable TCP [20] flow, or 5) a normal UDP flow. We assume that the maximum data rate of a streaming flow is 30 Mbps.

Figure 1 (a) shows the average throughput of a single streaming which adopts TFRC/TCP-Vegas/HSTCP/Scalable-TCP. The maximum average throughput of a TCP-Vegas flow under the RTTs becomes about 25 Mbps. The maximum average throughputs of both a HSTCP and Scalable TCP flow become around 30 Mbps. Because of losses caused by short-lived TCP flows, each of these TCP flows causes the regulations and fluctuations of the data transmission rate. Since these behaviors lead to a severe degradation of video quality, these TCP protocols are not suitable for a high-performance real-time streaming application. A TFRC flow under the RTTs achieves the average throughputs between 5 and 10 Mbps. Despite reducing the transmission rate, as shown in Fig. 1 (b), the average data loss rates of a TFRC flow under the RTTs becomes more than 0.1%, which is not much different from that of normal UDP flow. In addition, a normal UDP flow does not have a negative impact on the time necessary for TCP flows to complete data transfer. In this context, a TFRC flow with relatively high available bandwidth tends to suffer from a degradation of streaming quality caused by packet losses that TCP flows induce.

2.2 Requirement

As described and shown in Section 2.1, high-performance real-time streaming using TFRC suffers from an unnecessary reduction in the data transmission rate (i.e., video quality) due to improper congestion control. Instead of using TFRC, an alternative protocol is indispensable for high-performance real-time streaming to preserve the highest data transmission rate and avoid a degradation of playback quality caused by packet losses in congested networks, especially when the physical network bandwidth is much larger than the total consumption bandwidth of high-performance streaming flows [*2]. From this point of view, it is important to protect playback quality from packet loss while executing effective congestion control to achieve TCP friendliness.

An application level forward error correction (AL-FEC) approach prevents retransmission delays by preventively adding redundancy packets to the streaming data flow. Thanks to this redundancy, a certain number of missing data packets can be recovered. Even if a certain degree of FEC redundancy is applied to a high-performance streaming flow with TFRC, the flow cannot improve the video quality in congested networks as described before. However, ascertaining and controlling optimal FEC redundancy at the highest data transmission rate is a real challenge, because 1) it is difficult for a sender to determine the packet loss pattern at each moment (as there is a feedback delay) and to predict the futural packet loss pattern, and 2) increasing FEC redundancy may disturb both streaming and competing flows such as TCP when the conditions of competing flows are sensitively oscillated in the network. The following factors are important when enhancing FEC redundancy.

- Because failure to appropriately adjust FEC redundancy (such that FEC cannot recover lost data packets) will lead to non-recoverable data loss, FEC redundancy must be adjusted to optimize the recovery of lost data packets.
- Controlling FEC redundancy is not inconsequential to the behavior of other flows. It is important, therefore, to estimate the network conditions and the impact of FEC on the communication quality of other flows while controlling FEC redundancy.

3. Design of Dynamic Probing FEC Mechanism

In this section, we propose the dynamic probing forward error correction (DP-FEC) mechanism to satisfy the requirements aforementioned. According to the network condition, the mechanism changes "FEC window size" (packets) to make a packet loss tolerance, while considering the impact of FEC on the performances of competing TCP flows.

3.1 Overview

Figure 2 illustrates the DP-FEC overview. DP-FEC operates mainly at the sender. A sender transmits data and FEC repair packets over a real-time transport protocol (RTP) [3] carried on top of the Internet Protocol (IP) and UDP. The DP-FEC collects the feedback information transmitted over the real-time transport control protocol (RTCP) delivered by a receiver, then adjusts the degree of FEC redundancy using feedback information about packet losses. The data transmission rate, which depends on the video format preliminarily assigned, is maintained during transmission.

DP-FEC leverages an application level forward error correction (AL-FEC). In AL-FEC, $n - k$ repair packets are added to a block of k data packets. We consider maximum distance separable (MDS) codes such as Reed-Solomon codes [4] which can recover all of the missing packets from any set of exactly k packets. Here, we define the number of repair packets as the "FEC window size," indicated by "$n - k$". If the code parameters, the FEC window size and n, are appropriately set in the event of packet losses, a receiver may recover all of the missing data packets within a block. DP-FEC changes the FEC window size in a fixed FEC encoding block length n during transmission. Furthermore, all of the generated data packets are stored once in the FEC encoding block buffer. Depending on the FEC window size, the number of data packets stored in the FEC encoding block buffer will vary.

DP-FEC is designed as an end-to-end model in consideration of the following criteria:

(1) Utilizing network resources effectively, the FEC window size during data transmission is increased to make a packet loss tolerance as high as possible. At the same time, network estimation for congestion control can be conducted.
(2) In accordance with the network estimation, DP-FEC adjusts the FEC window size to minimize the impact of FEC on the performances of competing TCP flows.

3.2 DP-FEC Algorithm

The DP-FEC algorithm consists of three components: 1) the congestion estimation function, 2) repeating congestion estimation, and 3) the FEC window size adjustment function. Next, we describe the algorithm in detail and the parameters used.

3.2.1 Congestion Estimation Function

The decision to adjust the FEC window size can be summarized as follows:

- If no congestion is sensed, the FEC window size is increased except when the FEC window size is already at its maximum.
- If congestion is sensed, the FEC window size is decreased except when the FEC window size is at its minimum.

The DP-FEC recognizes above a certain degree of FEC impact on competing TCP performances as an indicator of network congestion. This means that the DP-FEC does not just use packet loss as a congestion indicator, but allows a certain level of packet loss during transmission. Here, we assume that a DP-FEC flow competes with TCP flows in the same bottleneck link, and define FEC impact as a ratio of FEC_bw to TCP_abw; FEC_bw is the consumption bandwidth of FEC repair packets that a DP-FEC flow adds per unit time, and TCP_abw is the available bandwidth that existing TCP flows can utilize per unit time. Thus, in competition with N TCP flows, the change in the degree of FEC impact (ΔFEC_impact) caused by added ΔFEC_bw is given by:

$$\begin{aligned}\Delta FEC\_impact &= (\Delta FEC\_bw/TCP\_abw)\\ &= (\Delta FEC\_bw/N)/(TCP\_abw/N)\\ &= (\Delta FEC\_bw/N)/(TCP1\_bw) \qquad (1)\end{aligned}$$

[*2] When the physical bandwidth is notably less than the total consumption bandwidth of high-performance streaming flows, end-users should reduce the sending data rate at the cost of video quality. In this study, we do not assume such a situation.

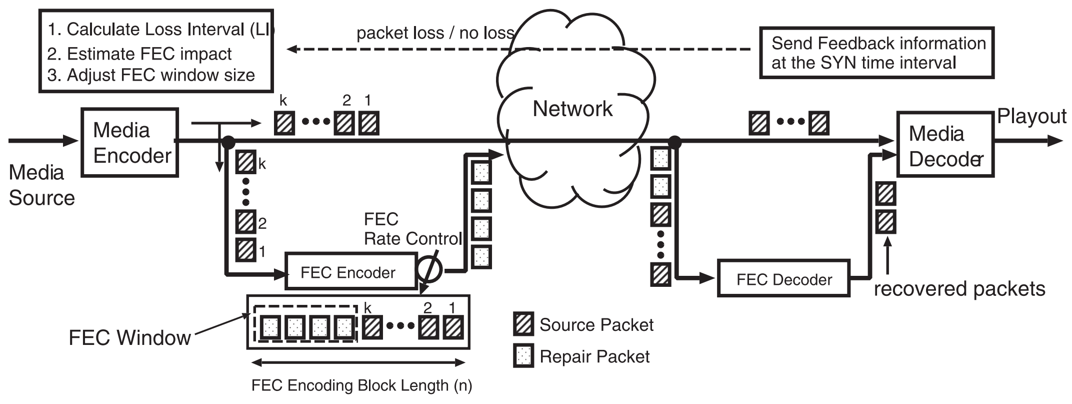


Fig. 2 The DP-FEC Overview.

where $TCP1\_bw$ is the data bandwidth that one TCP flow consumes. According to $TCP1\_bw$ and N, the FEC impact on the TCP performances varies. $TCP1\_bw$ can be estimated as follows [10]:

$$TCP1\_bw = \frac{PacketSize}{RTT\sqrt{\frac{2p}{3}} + RTO\left[3\sqrt{\frac{3p}{8}}p(1+32p^2)\right]} \quad (2)$$

where $PacketSize$ is the size of sending packet (bytes); p is the loss event rate; RTT is the Round Trip Time (sec); RTO is the retransmission timeout value (sec). If we can know N, ΔFEC_impact can be estimated. However, N changes from moment to moment, and it is naturally hard for an end system on an end-to-end model to precisely know N at the right time.

DP-FEC approximates $\Delta FEC\_bw/N$ by successively changing FEC_bw and observing the loss interval (LI) defined as the interval of time between packet loss events. An occurrence of more than one packet loss within a specific time is recognized as a single packet loss event. In order to approximate $\Delta FEC\_bw/N$, we now focus on the steady-state behavior of one TCP flow adopting the well-deployed AIMD algorithm [30] known as a mechanism for the remarkable stability of the Internet [11]. We assume that when a TCP flow observes more than one packet loss event during an RTT that results in halving the congestion window size, a DP-FEC flow also observes the same packet loss event. Since the TCP flow adopts the AIMD algorithm, LI can be approximated as follows:

$$LI \approx \frac{(TCP\_abw/N) * RTT^2}{2 * MSS} \quad (3)$$

where MSS is the TCP maximum segment size (bytes). When DP-FEC adds ΔFEC_bw, ΔLI can be approximated using Eq. (3), as follows:

$$\begin{aligned}\Delta LI &\approx \frac{(TCP\_abw/N) * RTT^2}{2 * MSS} \\ &\quad - \frac{\{(TCP\_abw - \Delta FEC\_bw)/N\} * RTT^2}{2 * MSS} \\ &= (\Delta FEC\_bw/N) * \frac{RTT^2}{2 * MSS} \quad (4)\end{aligned}$$

Using Eqs. (1), (2) and (4), we can approximate ΔFEC_impact as follows:

$$\Delta FEC\_impact \approx \frac{\Delta LI}{TCP1\_bw} * \frac{2 * MSS}{RTT^2} \quad (5)$$

DP-FEC estimates ΔFEC_impact using an observed ΔLI. An observation of a larger value of ΔLI implies that the increasing FEC window size has a more negative impact on TCP performances by reducing TCP_abw. DP-FEC recognizes network congestion when an estimated ΔFEC_impact exceeds *Threshold FEC Impact*. The value of *Threshold FEC Impact* in DP-FEC is set to 0.1. Since an observed ΔLI is influenced by packet loss patterns that vary according to ever-changing network conditions (e.g., the number of competing TCP flows and the available bandwidth), precisely estimating ΔFEC_impact is naturally difficult. Therefore, by successively observing ΔLI while changing the FEC window size, DP-FEC evaluates whether ΔFEC_impact exceeds *Threshold FEC Impact* or not. Although this strategy leads to low responsiveness to a change in network conditions, DP-FEC can effectively utilize network resources by adding FEC repair packets while considering the impact of FEC on TCP flows. The decision frequency is described in next Section 3.2.2.

As Eq. (5) shows, the estimation of ΔFEC_impact depends largely on RTT that competing TCP flows have, since TCP performance generally relies on RTT. If the RTT value is large, DP-FEC tends to conservatively behave and keep a small FEC window size even in slightly congested networks. This situation causes a number of non-recovered data packets without effectively utilizing network resources. To avoid such a behavior, DP-FEC estimates ΔFEC_impact assuming that all competing TCP flows have $RTT = 0.01$ (sec). Thus, in competition with TCP flows with $RTT > 0.01$, the DP-FEC tends to keep a larger FEC window size due to the underestimation of ΔFEC_impact.

3.2.2 Repeating Congestion Estimation

A DP-FEC receiver sends feedback information at constant time interval (SYN time), which denotes whether packet loss happened or not during the SYN time. Multiple packet losses in the same SYN time are considered as a single loss event. As described before, DP-FEC assumes that 1) both a DP-FEC flow and TCP flow observe the same packet loss event during an RTT and 2) all competing TCP flows have $RTT = 0.01$ (sec). Thus, the value of SYN time in DP-FEC is set to 0.01 (sec). A sender receives feedback information at the SYN time interval, and calculates the sample loss interval ($SampleLI$), the minimum value of which is equivalent to 0.01 (sec). Using the Exponential Weighted Moving Average (EWMA), current LI is calculated as follows: $LI = \alpha * LI + (1 - \alpha) * SampleLI$. DP-FEC uses the smoothing factor $\alpha = 0.9$, and estimates ΔFEC_impact with the observed variation of *LI* (ΔLI) at the constant interval (*ResTime*). The *ResTime* relates to the tradeoff between the aggressiveness (i.e., increasing rate of FEC window size) and estimation accuracy of *FEC_impact* (i.e., TCP-friendliness). In an ever-changing network condition, the appropriate setting of *ResTime* value is quite difficult. In this paper, we set *ResTime* to 0.16 (sec) based on our comprehensive simulation results as of now.

Figure 3 shows the relation between the current *LI* and maximum acceptable ΔLI, where MSS=1,460 (bytes), RTT=0.01 (sec). If an observed ΔLI exceeds the maximum acceptable ΔLI, a sender recognizes an occurrence of network congestion (i.e., $\Delta FEC\_impact > 0.1$). Note that if a network condition drastically becomes worse (e.g., due to an occurrence of bursty traffic), *LI* severely decreases. If this situation continues, DP-FEC needs to decrease the FEC window size in order to avoid congestion collapse. Therefore, DP-FEC also recognizes a network congestion when *LI* becomes less than the minimum value (*MinLI*). Every time a sender receives feedback information, the current *LI* is calculated and judged to be below *MinLI*. When DP-FEC uses an extremely-low value of *MinLI*, the attainable FEC window size becomes large even in highly congested networks due to the slow response to the network congestion. Such a situation causes a significant degradation of the performances of competing TCP flows. To avoid it, we set MinLI to 0.035 (sec) based on our simulation experiments as of now.

3.2.3 FEC Window Size Adjustment Function

DP-FEC recognizes network congestion when estimated ΔFEC_impact exceeds *Threshold FEC Impact* or current *LI* falls below *MinLI*. In DP-FEC, AIMD algorithm is used to adjust the FEC window size. FEC window size (*Fwnd*(*t*)) in packets is given by:

$$Fwnd(t) = \begin{cases} Fwnd(t-1)+1 & per\ ResTime \\ \lfloor \beta Fwnd(t-1) \rfloor & per\ congestion \end{cases} \tag{6}$$

During no congestion, *Fwnd* is linearly increased. During congestion, DP-FEC immediately decreases *Fwnd* using multiplicative factor β. DP-FEC with a low value of β tends to fail to recover lost data packets in continuously congested networks, because the attainable FEC window size becomes small. Since 1) DP-FEC should try to recover data loss as much as possible, and 2) by estimating and controlling ΔFEC_impact, DP-FEC can avoid a negative impact on TCP performances in network conditions where added FEC redundancy significantly disturbs TCP performances, DP-FEC sets β to a relatively high value, 0.8 (i.e., more than 0.5).

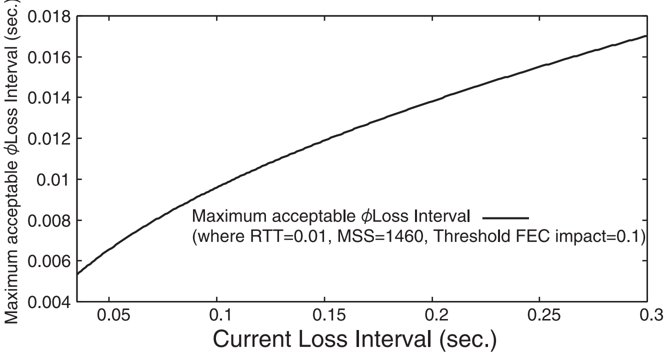


Fig. 3 Relation between the current calculated loss interval (*LI*) and maximum acceptable variance of loss interval (ΔLI).

4. Evaluation

4.1 Simulation Setup and Performance Metrics

Using an NS-2 simulator extended with DP-FEC module, we performed experiments using the dumbbell topology shown in **Fig. 4**. The bandwidth and propagation delay of the bottleneck link between *B*1 and *B*2 were set to 1 Gbps and 10 ms, respectively. A drop-tail queuing was used at the router, and the queue length size in packets was set to 128. Each sender and receiver were connected to the bottleneck link through the 1 Gbps access link with a propagation delay of 1 ms. The packet size was set to 1,500 bytes for all connections. Each simulation ran for about 120 seconds.

In all simulation experiments, a DP-FEC sender takes the same algorithm for sending data packets. It transmits smoothed data packets at a rate of 30 Mbps. As a practical matter, both the FEC encoding block length and the maximum FEC window size should be determined by the acceptable FEC encoding/decoding time for the streaming application. Here, we set the FEC encoding block length to 127 and the maximum FEC window size to 63, modeling a real environment.

As the DP-FEC performance metrics, we observed 1) the average residual data loss rate of DP-FEC flows (i.e., the rate of non-recovered data packet) and 2) the average throughput of competing TCP flows. The metric of the average residual data loss rate of DP-FEC flows was compared with those observed when we used a normal UDP flow at a rate of 30 Mbps or a TFRC flow [10] under the same network condition. As TCP friendliness index (*TFindex*), we use the result of the average throughput of TCP flows observed when they compete with TFRC flows, and define *TFindex* as follows:

$$TFindex = \frac{TCPthuput\_Target}{TCPthuput\_TFRC} \tag{7}$$

where *TCPthuput_Target* is the result of the average throughput of TCP flows competing with normal UDP flows or DP-FEC flows; *TCPthuput_TFRC* is the result of the average throughput of TCP flows competing with TFRC flows under the same network condition.

We used two types of TCP flows using a NewReno, 1) Short-lived TCP flows and 2) Long-lived TCP flows. Short-lived TCP

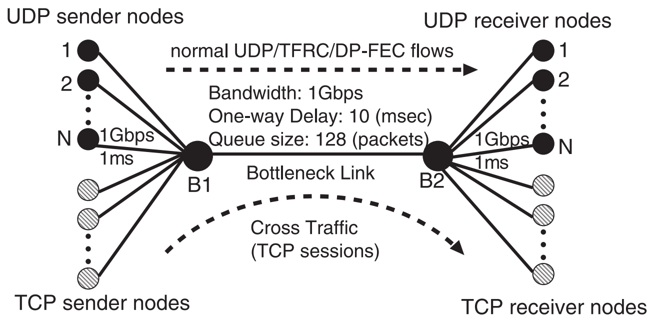
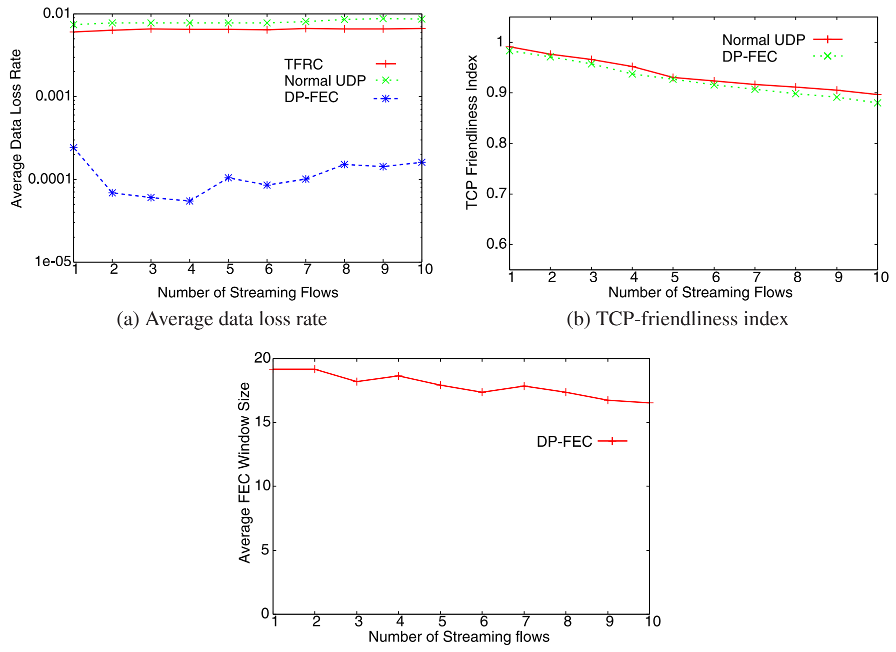


Fig. 4 Simulation network model.



(a) Average data loss rate

(b) TCP-friendliness index

(c) Average FEC window size

Fig. 5 DP-FEC Performances and average FEC window size under 25% load of TCP flows.

flows arrive at the bottleneck link, as a Poisson process with an average rate of r_tcp flows per second. The size of each TCP flow follows a Pareto distribution with an average of s_tcp packets and shape parameter 1.5. Here, we define the load of short-lived TCP flows as $\rho\_tcp = r\_tcp * s\_tcp$. Long-lived TCP flows are persistent in the network.

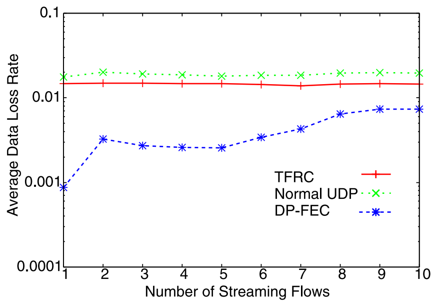
4.2 Competition with Only Short-Lived TCP Flows

We evaluated the DP-FEC performance in competition with only short-lived TCP flows. The load of short-lived TCP flows (ρ_tcp) was set to 25%, 50% and 75% of the bottleneck link capacity. In addition, the number of UDP streaming flows (i.e., TFRC, normal UDP or DP-FEC flows) was changed from 1 to 10.

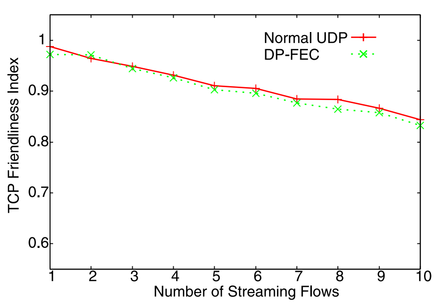
As shown in **Fig. 5** (a), the average data loss rates of DP-FEC flows under 25% load of TCP flows are considerably improved by added FEC redundancy, compared to those of TFRC or normal UDP flows. In addition, Fig. 5 (b) shows that the TCP friendliness indexes of both normal UDP and DP-FEC flows are maintained at more than approximately 0.9, which means that DP-FEC flows add a well-coordinated degree of FEC redundancy by effectively utilizing network resources that competing TCP flows cannot consume. In Fig. 5 (c), we can see that the FEC window sizes are almost the same (between 17 and 20) regardless of the number of competing DP-FEC flows, because FEC impacts that each DP-FEC flow estimates during transmission are not largely different (i.e., there is little difference in the available bandwidth for TCP flows in each test). Because of an occurrence of packet losses caused by a behavior in slow start phase of generated short-lived TCP flow, the average FEC window size is suppressed to below 20. TFRC flows in each test also observe packet losses caused by a slow start phase of generated short-lived TCP flow, and reduce the data transmission rates to about 6 Mbps. As opposed to a TFRC flow, a DP-FEC flow suppresses the average data loss rate and preserves the highest data transmission rate while achieving high TCP friendliness index.

When ρ_tcp is 50% of the bottleneck link capacity, the average data loss rates of TFRC and normal UDP flows becomes more than 1%, as shown in **Fig. 6** (a). On the other hand, owing to added FEC redundancy, the average data loss rates of DP-FEC flows are suppressed to below 1%. In addition, Fig. 6 (b) shows that although the TCP friendliness indexes of both normal UDP and DP-FEC flows decrease with an increase in the number of streaming flows, DP-FEC flows in each test retains more than approximately 0.85% of the TCP friendliness index. In Fig. 6 (c), we can see that as the number of DP-FEC flows increases, the average FEC window size gradually decreases to about 9. This is because, under 50% load of TCP flows, DP-FEC flows tend to experience continuous packet loss and estimate higher FEC impact with an increase in the number of DP-FEC flows. The continuous packet loss leads to an increase in average data packet loss rate, as shown in Fig. 6 (a).

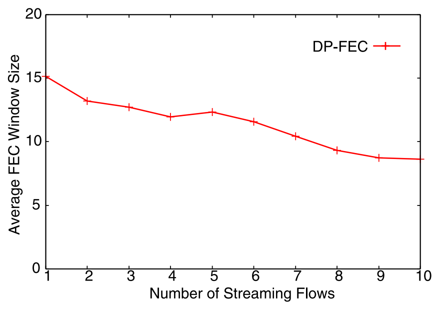
When ρ_tcp is 75% of the bottleneck link capacity, the average loss rates of DP-FEC flows are slightly improved except when the number of DP-FEC flows is more than 8, as shown in **Fig. 7** (a). In Fig. 7 (b), we can see that as the number of streaming flows increases, the TCP friendliness indexes of both flows of Normal UDP and DP-FEC severely decrease because of a reduction in the available bandwidth for heavy short-lived TCP traffic. Due to the continuous packet loss caused by heavy short-lived TCP traffic, the average FEC window sizes are suppressed to between 5 and 10, as shown in Fig. 7 (c). Since DP-FEC is designed to minimize FEC impact defined as a ratio of the consumption bandwidth of FEC repair packets to the available bandwidth for TCP flows, the TCP friendliness indexes of DP-FEC flows become approximately the same as those of normal UDP. Therefore, in a situation



(a) Average data loss rate

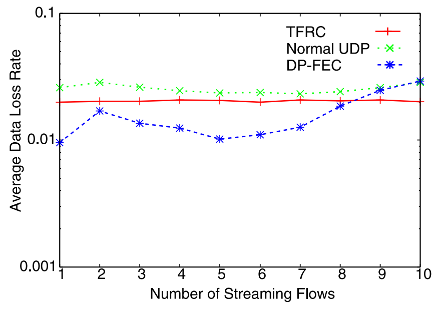


(b) TCP-friendliness index

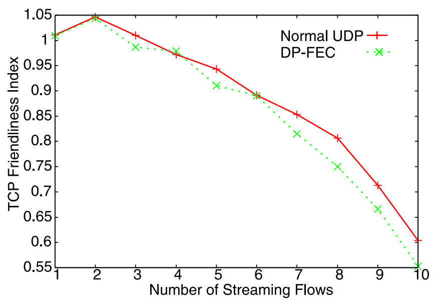


(c) Average FEC window size

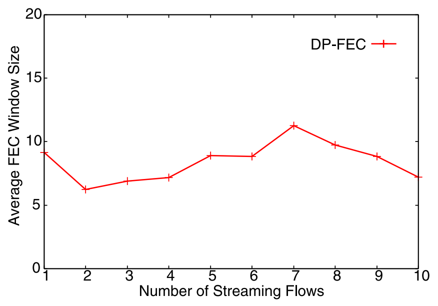
Fig. 6 DP-FEC Performances and average FEC window size under 50% load of TCP flows.



(a) Average data loss rate



(b) TCP-friendliness index



(c) Average FEC window size

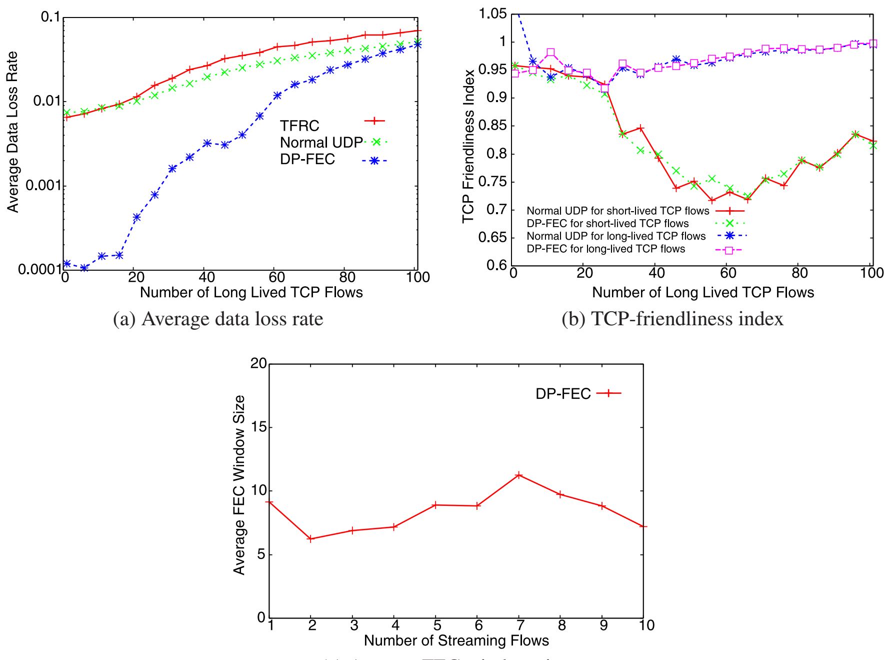
Fig. 7 DP-FEC Performances and average FEC window size under 75% load of TCP flows.

in which a normal UDP flow severely decreases the TCP friendliness index, a DP-FEC cannot retain a higher TCP friendliness index.

4.3 Competition with Both Short-Lived TCP Flows and Long-Lived TCP Flows

We evaluated the DP-FEC performance in competition with both short-lived TCP flows and long-lived TCP flows. The load of short-lived TCP flows (ρ_tcp) was set to 25% of the bottleneck link capacity, and the number of long-lived TCP flows was changed from 1 to 101. The number of UDP streaming flows was set to 3.

Figure 8 (a) shows that whereas the average data loss rates of TFRC and normal UDP flows become more than 1% in competition with more than 20 of long-lived TCP flows, those of DP-FEC flows are suppressed to below 1% in competition with less than 60 of long-lived TCP flows. As the number of long-lived TCP flows increases, the average data loss rates of DP-FEC flows also increase and approach those of TFRC and normal UDP flows. This is because many long-lived TCP flows probe available bandwidth during the congestion avoidance phase and incur more continuous packet loss. Due to the continuous packet loss, DP-FEC flows recognize higher FEC impact, resulting in the reduction of average FEC window sizes, as shown in Fig. 8 (c). However, the aver-



(a) Average data loss rate (b) TCP-friendliness index

(c) Average FEC window size

Fig. 8 DP-FEC Performances and average FEC window size in competition with long-lived TCP flows, under 25% load of TCP flows.

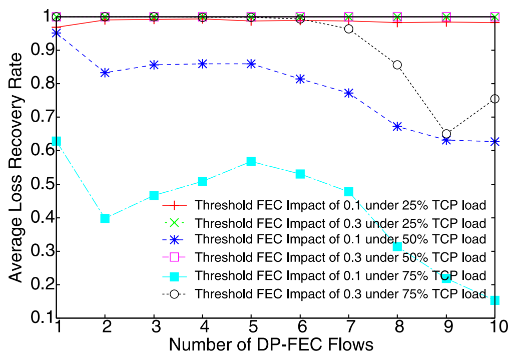
age FEC window sizes are kept between 15 and 20 when the number of long-lived TCP flows is less than or equal to 60. This condition is almost the same as in the case of competition with only short-lived TCP flows causing 25% load, because a DP-FEC flow recognizes that there is a certain level of available bandwidth by observing the interval between packet loss events while adjusting the FEC window size (i.e., the available bandwidth for TCP flows is not largely different despite increasing in FEC window size). Figure 8 (b) shows the TCP friendliness indexes of normal UDP flows and DP-FEC flows. The TCP friendliness index is distinguished by the type of TCP flow. Although DP-FEC flows retain the average FEC window size between 15 and 20 in competition with less than 60 of long-lived TCP flows, the TCP friendliness indexes for long-lived TCP flows always become more than 0.9. Whereas a TFRC flow reduces the data transmission rate to below about 6 Mbps under the influence of packet loss conditions, a DP-FEC flow utilizes network resources effectively by adding FEC repair packets which contribute to the recovery of data packet losses. On the other hand, the TCP friendliness indexes of both normal UDP and DP-FEC flows for short-lived TCP flows become more than 0.9 when the number of long-lived TCP flows is below 25. When the number of long-lived TCP flows becomes more than 25, the TCP friendliness indexes for short-lived TCP flows becomes between 0.7 and 0.9. This is because the total consumption bandwidth of long-lived TCP flows increases, which deteriorates the performances of short-lived TCP flows. However, as described in Section 4.2, the TCP friendliness indexes of DP-FEC flows for short-lived TCP flows become approximately the same as those of normal UDP, since DP-FEC flows try to suppress the impact of FEC on the TCP performances.

4.4 Effect of the Value of Threshold FEC Impact

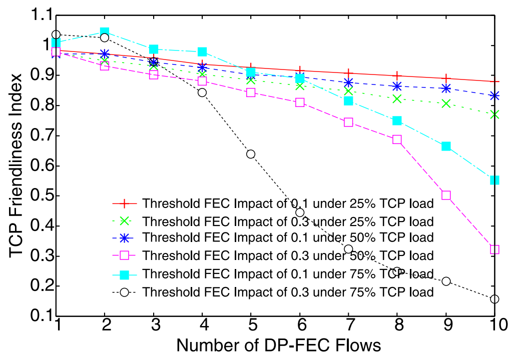
The value of *Threshold FEC Impact* (the default value is 0.1) is considerably important for DP-FEC to recognize network congestion and adjust the FEC window size. Thus, we evaluated the performances of DP-FEC with *Threshold FEC Impact* of 0.3 under TCP load of 25%/50%/75%, and compared the results with those of DP-FEC with *Threshold FEC Impact* of the default value.

Figure 9 (a) shows the average loss recovery rates of DP-FEC flows (the cases of *Threshold FEC Impact* of 0.1/0.3) under 25%/50%/75% load of TCP flows. The loss recovery rate is defined as the ratio of the number of the recovered data packets to that of the lost data packets in the network. We can see that under 25% load of TCP flows, DP-FEC flows with *Threshold FEC Impact* of 0.1 maintain the average loss recovery rates of more than 0.98. However, under both 50% and 75% load of TCP flows, the average loss recovery rates decrease because the attainable FEC window size of each DP-FEC flow tends to become small as described in Section 4.2. On the other hand, the average loss recovery rates of DP-FEC flows with *Threshold FEC Impact* of 0.3 almost become 100% except when the number of DP-FEC flows under 75% load of TCP flows is more than 6. This is because a DP-FEC flow with *Threshold FEC Impact* of 0.3 keeps larger FEC window size than that of a DP-FEC flow with *Threshold FEC Impact* of 0.1.

Figure 9 (b) shows the corresponding TCP friendliness indexes of DP-FEC flows. When the number of competing DP-FEC flows is the same, the TCP friendliness indexes of DP-FEC flows with *Threshold FEC Impact* of 0.3 becomes lower than those of DP-FEC flows with *Threshold FEC Impact* of 0.1. Under 25% load of TCP flows, the difference of TCP friendliness indexes is not large regardless of the number of DP-FEC flows. However, as the number of DP-FEC flows increases under 50% or 75% load of TCP flows, the difference becomes large (i.e., the TCP friendliness indexes of DP-FEC flows with *Threshold FEC Impact* of



(a) Average loss recovery rate



(b) TCP-friendliness index

Fig. 9 Average loss recovery rates and TCP friendliness indexes of DP-FEC flows (the cases of *Threshold FEC Impact* of 0.1/0.3 under TCP load of 25%/50%/75%).

0.3 largely decrease). This is because, since DP-FEC flows with *Threshold FEC Impact* of 0.3 tends to keep a larger FEC window size, the total bandwidth of FEC repair packets increases considerably. Since the value of *Threshold FEC Impact* relates to the tradeoff between efficiency (i.e., loss recovery rate) and TCP friendliness, it is of greater importance for DP-FEC to appropriately set the value.

5. Related Work

Many congestion control mechanisms for real-time streaming have been proposed and analyzed. Rejaie et al. [1] developed a rate-based congestion control mechanism that employs an additive-increase, multiplicative-decrease (AIMD) algorithm to achieve TCP-like behavior. The datagram congestion control protocol (DCCP) [26] was proposed by Kohler et al. to provide TCP-friendly rate control [10], [12]. The mechanism function is designed to behave fairly with coexisting TCP flows in terms of the consumption bandwidth. Papadimitriou et al. have proposed a rate control mechanism from the perspective of inter-protocol fairness [11]. Using a packet loss as a congestion indicator, this mechanism decreases the data transmission rate by a previously determined degree. If no congestion is sensed, it periodically increases the data transmission rate in incremental steps. Feng et al. [27] defined the new TCP friendliness necessary to achieve a desired constant data rate while maintaining fairness with coexisting TCP flows. However, because these mechanisms utilize packet loss as a congestion indicator and severely reduce the data transmission rate in response, they are not suitable for high-quality real-time video streaming that needs to maintain a higher data rate and avoid a degradation of playback quality, as described in Section 2.

Various numerical studies, using actual experimental measurement or analytical models, have evaluated the effectiveness of FEC. Bolot et al. [21] measured audio packets transmitted via the Internet, and analyzed the packet loss patterns. The results suggested that FEC is effective for low bandwidth streaming, such as an audio application, even when the network conditions are unstable because of network congestion. However, Altman et al. [25] proposed a detailed queuing analysis based on a ballot theorem, and concluded that FEC caused only a slight improvement in the audio streaming quality under any amount of redundant FEC data. In Ref. [6], an analysis of FEC effectiveness in ATM networks indicated that performance gains quickly diminish when all traffic sources employ FEC and the number of sources increases. Shacham et al. [7] used both analytic and simulation models to evaluate a situation using redundant parity packets, residual packet-loss rates after FEC decoding were reduced by up to three orders of magnitude. However, Xunqi et al. [8] indicated that the simplified analysis in Ref. [7] overestimated the performance of FEC because it assumed that the packet loss process is independent. As an alternative approach, Xunqi et al. [8] proposed a recursive algorithm based on the algorithm proposed by Cidon et al. [9] to compute block error density, and analyzed FEC performance in various FEC parameters (i.e., the encoding length, code rate, and interleaving depth). They concluded that an interleaving method increases FEC effectiveness for real-time streaming over congested networks. Although these numerical results vary depending on the loss model that is used or the testbed network, they provide insights into FEC recovery capability that can be applied to audio and video streaming.

A number of mechanisms for adjusting FEC redundancy have been proposed. Bolot et al. [31] mentioned that the congestion losses were modeled with a two-state Gilbert model, and the proposed scheme determined the FEC redundancy together with a TCP-friendly rate control scheme. Wu et al. [22] described that bandwidth constraints derived by the TFRC equation were used to optimally adjust FEC redundancy and the data transmission rate to maximize the video and playback quality. Seferoglu et al. [17] proposed TFRC with FEC to deal with packet losses induced by competing TCP flows. The mechanism utilizes the correlation between packet losses and the estimated RTT fluctuation as a network indicator. Moreover, they provided a rate-distortion optimized way to decide the best allocation of the available TFRC rate between source and FEC packets, and performed a significantly extended performance evaluation [18]. However, such a delay-based control which susceptibly reacts to RTT variations leads to a severe quality degradation, especially in high bandwidth paths [15], [16]. In addition, because these proposed mechanisms depend largely on the TFRC throughput equation, they do not effectively utilize network resources, which results in a failure to maintain higher video quality. In contrast, the approach of our proposed DP-FEC does not rely on TFRC rate to avoid a degradation of video quality caused by the regulation of the sending rate of source packets. We previously proposed an adaptive rate control mechanism that dynamically adjusts both the FEC redundancy and the data transmission rate [29]. It is an effective technique for a single stream. However, it does not cooperate with other competing flows. This type of self-organized method adversely increases traffic congestion and disturbs other communication qualities. However, our proposed DP-FEC attempts to utilize the network resource effectively to optimize the recovery of lost data while minimizing the impact of FEC on other competing flows.

6. Conclusion and Future Work

In this paper, we proposed the DP-FEC that effectively achieves streaming quality for a high-performance real-time streaming. This mechanism tolerates packet loss in the network through the adjustments of the FEC window size while examining network congestion. By successively observing variation in the intervals between packet loss events, DP-FEC effectively utilizes network resources and adjusts the degree of FEC redundancy while minimizing the impact of FEC on the performance of competing TCP flows. We verified the efficiency of DP-FEC using an NS-2 simulator, and recognized that DP-FEC retains high streaming quality.

In our future work, we will make more investigation to optimally set a parameter such as *Threshold FEC impact* according to network conditions. Then, we will evaluate the DP-FEC mechanism competing with more high-performance streaming and TCP flows in complex and higher bandwidth networks. This evaluation will improve the DP-FEC algorithm, and then accelerate deployment of high-quality streaming applications over future heterogeneous networks.

Reference

[1] Rejaie, R., Handley, M. and Estrin, D.: RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet, *Proc. IEEE INFOCOM '99*, New York, USA (Mar. 1999).

[2] Nakashima, N., Okamura, K., Hahm, J.S., Kim, Y.W., Mizushima, H., Tatsumi, H., Moon, B.I., Han, H.S., Park, Y.J., Lee, J.H., Youm, S.K., Kang, C.H. and Shimizu, S.: Telemedicine with digital video transport system in Asia-Pacific area, *Proc. 19th International Conference on Advanced Information Networking and Applications* (Mar. 2005).

[3] Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V.: RTP: A Transport Protocol for Real-Time Applications, RFC 3550 (July 2003).

[4] Rizzo, L.: Effective erasure codes for reliable computer communication protocols, *Proc. ACM Comput. Commun*, Vol.27, No.2, pp.24–36 (Apr. 1997).

[5] Ogawa, A., Kobayashi, K., Sugiura, K., Nakamura, O. and Murai, J.: Design and implementation of DV based video over RTP, *Proc. International Packet Video Workshop* (*PV 2000*) (May 2000).

[6] Biersack, E.: Packet Recovery in High-Speed networks using coding and buffer management, *Proc. ACM SIGCOMM'90* (Nov. 1992).

[7] Shacham, N. and Mckenney, P.: Packet recovery in high-speed networks using coding and buffer management, *Proc. IEEE INFOCOM'90*, San Francisco, CA (June 1990).

[8] Xunqi, Y., Modestino, J.W., Kurceren, R. and Chan, Y.S.: A Model-Based Approach to Evaluation of the Efficacy of FEC coding in Combating Network Packet Losses, *IEEE/ACM Trans. on Networking*, Vol.16, No.3, pp.628–641 (June 2008).

[9] Cidon, I., Khamisy, A. and Sidi, M.: Analysis of packet loss processes in high-speed networks, *IEEE Trans. Inf. Theory*, Vol.39, No.1, pp.98–108 (1993).

[10] Floyd, S., Handley, M., Padhye, J. and Widmer, J.: Equation-based congestion control for unicast applications, *Proc. ACM Comput. Commun*, Vol.30, No.4, pp.43–56 (Oct. 2000).

[11] Papadimitriou, P. and Tsaoussidis, V.: A rate control scheme for adaptive video streaming over the Internet, *Proc. IEEE ICC '07*, Glasgow, Scotland (June 2007).

[12] Floyd, S., Handley, M., Padhye, J. and Widmer, J.: TCP Friendly Rate Control (TFRC): Protocol Specification, RFC 5348 (Sep. 2008).

[13] Brakmo, L.S., O'Malley, S.W. and Peterson, L.L.: TCP Vegas: new techniques for congestion detection and avoidance, *Proc. ACM Comput. Commun*, Vol.24, No.4, pp.24–35 (1994).

[14] Wei, D.X., Jin, C., Low, S.H. and Hegde, S.: FAST TCP: Motivation, architecture, algorithms, performance, *IEEE/ACM Trans. on Networking*, Vol.14, No.6, pp.1246–1259 (2006).

[15] Prasad, S., Jain, M. and Dovrolis, C.: On the effectiveness of delay-based congestion avoidance, *PFLDnet Workshop* (Feb. 2004).

[16] Martin, J., Nilsson, A. and Rhee, L.: Delay-based congestion avoidance for TCP, *IEEE/ACM Trans. on Networking*, Vol.11, No.3, pp.356–369 (2003).

[17] Seferoglu, H., Kozat, U.C., Civanlar, M.R. and Kempf, J.: Congestion state-based dynamic FEC algorithm for media friendly transport layer, *Proc. International Packet Video Workshop* (*PV 2009*) (May 2009).

[18] Seferoglu, H., Markopoulou, A., Kozat, U.C., Civanlar, M.R. and Kempf, J.: Dynamic FEC Algorithms for TFRC Flows, *IEEE Trans. on Multimedia*, Vol.12, No.8, pp.869–885 (2010).

[19] Floyd, S.: Highspeed TCP for Large Congestion Window, IETF RFC3649 (2003).

[20] Kelly, T.: Scalable TCP: Improving Performance in High-speed Wide Area Networks, *PFLDnet Workshop* (Feb. 2004).

[21] Bolot, J.C., Crépin, H. and Vega-Garcia, A.: Analysis of Audio Packet Loss over Packet-Switched Networks, *Proc. ACM NOSSDAV '95*, New Hampshire, USA (Apr. 1995).

[22] Wu, H., Claypool, M. and Kinicki, R.: Adjusting forward error correction with quality scaling for streaming MPEG, *Proc. ACM NOSSDAV '05*, Washington, USA (June 2005).

[23] Bao, C., Li, X. and Jiang, J.: Scalable Application-Specific Measurement Framework for High Performance Network Video, *Proc. ACM NOSSDAV '07*, Urbana, Illinois USA (2007).

[24] Zhang, Y. and Loguinov, D.: Oscillations and Buffer Overflows in Video Streaming under Non-Negligible Queuing Delay, *Proc. ACM NOSSDAV '04*, Cork, Ireland (2004).

[25] Altman, E., Barakat, C. and Ramos, V.M.: Queuing analysis of simple FEC schemes for IP telephony, *Proc. IEEE INFOCOM* (Nov. 2001).

[26] Kohler, E., Handley, M. and Floyd, S.: Designing DCCP: Congestion control without reliability, *Proc. ACM SIGCOMM'06*, Piza, Italy (Sep. 2006).

[27] Feng, J. and Xu, L.: TCP-Friendly CBR-Like Rate Control, *Proc. IEEE ICNP* (Oct. 2008).

[28] Jin, C., Wei, D.X. and Low, S.H.: FAST TCP: Motivation, architecture, algorithms, performance, *IEEE Infocom'04*, Hongkong, China (Mar. 2004).

[29] Matsuzono, K., Sugiura, K. and Asaeda, H.: Adaptive rate control with dynamic FEC for real-time DV streaming, *Proc. IEEE Globecom '08*, New Orleans, USA (Dec. 2008).

[30] Chiu, D. and Jain, R.: Analysis of the increase and decrease algorithm for congestion avoidance in computer networks, *Comput. Netw. ISDN Syst. J.*, Vol.17, No.1, pp.1–14 (June 1989).

[31] Bolot, J.C., Fosse-Parisis, S. and Towsley, D.: Adaptive FEC-based error control for Internet telephony, *Proc. IEEE INFOCOM '99*, New York, USA (Mar. 1999).

Kazuhisa Matsuzono received his B.E from Keio University in 2005. He received his master's degree in 2007 from Graduate School of Media and Governance Keio University and is now a Ph.D. student in the same university. His current research interests include streaming protocol and its architecture.

Hitoshi Asaeda is Associate Professor of Graduate School of Media and Governance, Keio University. He received his Ph.D. in Media and Governance from Keio University. in 2006. From 1991 to 2001, he was with IBM Japan, Ltd. From 2001 to 2004, he was a research engineer specialist at INRIA Sophia Antipolis, France. His research interests are IP multicast routing architecture and its deployment, dynamic networks and streaming applications. He is a member of ACM, IEEE, IPSJ, and WIDE Project.

Jun Murai is Professor of Faculty of Environment and Information Studies, Keio University. He received his M.E. and Ph.D. in Computer Science from Keio University in 1981 and 1987 respectively. He was a director of Keio Research Institute at SFC, the president of Japan Network Information Center (JP-NIC), board director of ICANN. Adjunct Professor at Institute of Advances Studies, United Nation University. He also teaches computer network and computer communication.