Journal of Information Processing Vol.21 No.1 109–121 (Jan. 2013)

[DOI: 10.2197/ipsjjip.21.109]

Regular Paper

GENEVA: Streaming Control Algorithm Using Generalized Multiplicative-increase/additive-decrease

Kazuhisa Matsuzono[1,a)] Hitoshi Asaeda[1] Osamu Nakamura[2] Jun Murai[2]

Received: March 17, 2012, Accepted: October 10, 2012

Abstract: Motivated by the deployment of wide-area high-speed networks, we propose GENEVA, the streaming control algorithm using generalized multiplicative-increase/additive-decrease (GMIAD). Because current typical congestion controllers such as a TCP-friendly rate control prevent occurrences of network congestion reacting susceptibly to packet loss, it causes a significant degradation of streaming quality due to low-achieving throughput (i.e., lower throughput than the maximum throughput that a streaming flow requires in maximum audio/video quality) and data packet losses. GENEVA avoids this problem by allowing a streaming flow to maintain moderate network congestion while trying to recover lost data packets that other competing flows cause during the process of probing for available bandwidth. Using the GMIAD mechanism, the FEC window size (the degree of FEC redundancy per unit time) is adjusted to suppress occurrences of bursty packet loss, while trying to effectively utilize network resources that other competing flows cannot consume due to reductions in the transmission rate in response to packet loss. We describe the GENEVA algorithm and evaluate its effectiveness using an NS-2 simulator. The results show that GENEVA enables high-performance streaming flows to retain higher streaming quality under stable conditions while minimizing the adverse impact on competing TCP performance.

Keywords: real-time streaming, congestion control, forward error correction, GMIAD

1. Introduction

The Internet has been growing by increasing in both access links and backbone networks. According to the State of the Internet published by Akamai Technologies [1], 1) the global average peak connection speed, which implies Internet connection capacity, has been growing, and 2) the speeds in the top 5 countries exceed 30 Mbps. In fact, 1 Gbps broadband access (FTTH) services have been provided with reasonable price in Japan, and 10 Gbps services will be in widespread use in the near future. Thanks to the dissemination of wide-area high-speed networks [23], [24], global IP traffic has increased eightfold over the past 5 years, and it is expected that global IP traffic will reach zettabytes by the end of 2015 [2]. This condition encourages people to take advantage of high-quality and high-performance real-time streaming applications that consume a large amount of network bandwidth for data transmission (e.g., a single Digital Video (DV) stream requires 30 Mbps [9], [22]). Such applications will be in widespread use, for example, in interactive e-learning, at international symposiums, and in telemedicine [5]. Real-time streaming applications commonly rely on the unreliable transport services provided by UDP (a fast and lightweight protocol). In this study, we assume that these applications are implemented on top of UDP.

On the other hand, although each packet loss ratio measured for various regions has decreased due to the capacity of broadband connections, users still observe around 1% packet loss continuously [3]. Such a small amount of packet loss seriously degrades video quality. For example, as little as 3% MPEG packet loss can cause 30% of the frames to be undecodable [21]. The exponential increase in global IP traffic aggravates the adverse condition even in high-speed networks, because TCP flows (which accounts for more than 90% of the Internet traffic [31]) induce network congestion by examining maximum available bandwidth, and then leads to *bursty* packet loss [30]. Meanwhile, even if network bandwidth will further grow, such streaming quality degradation will not be fundamentally addressed. Rather, it is a formidable challenge to high-performance streaming applications seeking to optimize their quality without affecting competing traffic flows (e.g., a large amount of TCP flows) in an environment where both network connection speed and IP traffic flows have been growing.

As one of the major approaches for real-time streaming to deal with network congestion, TCP-friendly rate control [12], [14] has been well studied. 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 mechanism for a smooth data transmission rate [17], [28]. TFRC reduces the data transmission rate to prevent occurrences of network congestion reacting to packet loss, and increases the data transmission rate while probing for available bandwidth in the absence of packet loss. However, it is well known that in networks with high bandwidth-delay products (BDP), TCP is often found to fail to utilize network resources. BDP denotes the prod-

1 Graduate School of Media and Governance, Keio University, Fujisawa, Kanagawa 252–0882, Japan
2 Faculty of Environment and Information Studies, Keio University, Fujisawa, Kanagawa 252–0882, Japan
a) kazuhisa@sfc.wide.ad.jp

uct of a link capacity (bits/sec) and its end-to-end delay (sec). When applied to the context of the TCP, BDP can be considered as a measurement of the maximum amount of unacknowledged data sent to the link (i.e., the maximum amount of data carrying capacity of the link when there is no competing traffic). In addition, the possible lack of a buffer on routers and the existence of concurrent bursting flows prevent TCP flow from effectively utilizing network bandwidth [26]. TFRC in such conditions may unnecessarily force a high-performance streaming flow to reduce the data transmission rate at the expense of video quality. In networks where the physical bandwidth is relatively larger than the total consumption bandwidth of high-performance streaming flows, senders maintaining the highest data transmission rate would not make a severe impact on the congestion [27]. Thus, high-performance streaming flows using TFRC suffer from a significant degradation of streaming quality due to low-achieving throughput (i.e., lower throughput than the maximum throughput that a streaming flow requires in maximum audio/video quality) and bursty packet loss.

In this paper, we focus on the scenario of multiple coexisting flows along a high-speed network path (more than 1 Gbps), where both high-performance real-time streaming flows transmitting at a rate of several tens of Mbps and TCP flows are competing. We assume that round trip times (RTTs) the competing flows have in such an advanced broadband network environments are between about 10 and 200 ms [3]. The key assumption in our research is that network congestion as indicated by packet loss occurs mainly due to TCP flows, where the bottleneck link bandwidth is notably larger than the total consumption bandwidth of high-performance streaming flows sending data packets at the maximum rate (i.e., at the maximum audio/video quality). To tackle the problem that an existing congestion controller fails to maintain higher streaming quality for high-performance real-time applications in high BDP networks, we propose GENEVA, the streaming control algorithm using the generalized multiplicative-increase/additive-decrease (GMIAD). GENEVA also addresses the problem of an adverse effect on competing TCP performance and a degradation of streaming quality in DP-FEC of our previous work [29]. Although DP-FEC increases the degree of FEC redundancy for probing available bandwidth regardless of RTT conditions and data transmission rate, GENEVA using GMIAD mechanism considers them to adjust the degree of FEC redundancy. GENEVA provides a promising method for achieving higher streaming quality by adding redundant data packets while both effectively utilizing network resources and combating bursty packet losses through adjustment of the transmission rate using the GMIAD mechanism. GENEVA avoids low-achieving throughput by allowing the streaming flows to maintain moderate network congestion, and also considers the effect on the performance of other competing flows. The GMIAD mechanism adjusts the consumption bandwidth by changing the degree of redundant data packets to suppress bursty packet loss, which contributes to reliability and stability of the streaming playback quality. In addition, it tries to effectively utilize expected available bandwidth that competing TCP flows cannot consume due to reductions in the transmission rate in response to packet loss. We describe the GENEVA algorithm and evaluate its effectiveness using an NS-2 simulator. The results show that GENEVA enables high-performance streaming flows to retain higher streaming quality under stable conditions while minimizing the adverse impact on competing TCP performance.

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

2. Congestion Control for Video Streaming over IP

2.1 Background

Real-time streaming for high-quality video contents generally tries to keep pace with changes in network conditions. To escape from interruptions and stalling, streaming applications compensate for packet loss by various approaches whose suitability could be dependent on encoding techniques and communication environments. High-quality real-time video streaming has strict requirements on interactivity (i.e., low perceived latency) and packet loss (i.e., high perceived playback quality). To keep both interactivity and video quality as high as possible, high-quality real-time streaming prefers to use low video compression and high data transmission rate. However, for high-quality real-time streaming applications, an expected lower transmission rate by a traditional rate or congestion control algorithm often results in buffer underrun at the receiver side, which results in a degradation of playback quality. This is because the rate reduction by changing the encoding parameters at the sender side needs more time to encode/decode. It is therefore important to maintain the highest data transmission rate when there is fully available bandwidth.

2.2 Problem Statement

Most of the proposed congestion control mechanisms for real-time streaming applications [4], [12], [13], [17], [25], [28] try to achieve TCP-friendliness according to its definition; TFRC or Datagram Congestion Control Protocol (DCCP) [25] thus enables a streaming flow to maintain the desired smoothness of data transmission rate and fairness with coexisting TCP flows in the same network condition (i.e., the packet loss event rate and RTT). To maintain fairness with coexisting TCP flows, TFRC calculates the expected throughput of a competing TCP flow by using an equation as a function of loss event rate and RTT [14], and then adjusts the data transmission rate so as not to exceed the expected TCP throughput. The expected TCP throughput based on the equation becomes small as packet loss rate or RTT increases. However, because high-performance streaming flow keeping the maximum video quality continuously requires a relatively large amount of bandwidth that cannot be utilized by TCP in congested networks, the throughput regulation based on TCP-friendliness forces the streaming flow to reduce the data transmission rate at the expense of video quality (i.e., the transmission rate is adjusted by changing the parameters for encoding the video), which may cause low network utilization by giving excessive weight to TCP-

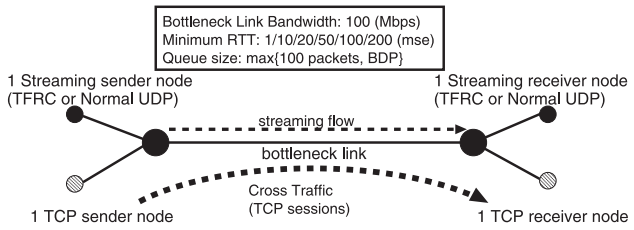


Fig. 1 The simulation topology (single bottleneck link).

friendliness. Video quality is increasingly reduced as the bandwidth delay product (BDP) increases, because TCP is inefficient under high BDP environments (i.e., the expected throughput of a TCP flow by TFRC tends to become small under high BDP environments). A high-performance TCP protocol (such as scalable TCP [36] and high-speed TCP [37]) 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). As another critical problem, since the window controls – competing TCP flows adopt – send data packets in bursts (and may have sent a large number of packets by the time the sender learns of an occurrence of network congestion), streaming flows suffer from bursty data packet losses (i.e., consecutively lost data packets). In addition, since a TFRC flow transmitting at a lower data rate than a maximum data rate increases the data transmission rate during the process of probing for available bandwidth and examines packet loss conditions as an indicator of network congestion, it often delays the action for rate control, where it leads to a situation of continuous or further data packet loss.

To clarify the specific problem of competition between a TFRC streaming flow and TCP flows, we used NS-2 to simulate a single bottleneck with a 100 Mbps link capacity. The network topology is shown in **Fig. 1**. Each sender and receiver were connected to the bottleneck link through the 1 Gbps access link with propagation delay of 0.1 ms. The two-way propagation delay was set to between 1 and 200 ms (1/10/20/50/100/200 ms). From here onwards, we call this delay minimum RTT (RTTmin). According to RTTmin settings, the propagation delay in the bottleneck link varies, and BDP is calculated using the RTTmin. The packet size was set to 1,500 bytes for all connections. A drop-tail queuing is used at the router in the bottleneck link, the queue size is set to max{100 (packets), BDP}. The queue size in packets in RTTmin = 1 ms is calculated as follows: $BDP = 100\,(\text{Mbps}) * 10^6 * 0.001\,(\text{sec})/(1500 * 8) = 8.33$ (packets). Since the queue size set to the BDP in RTTmin = 1 and 10 ms is less than 100 packets, the queue size was set to 100 packets, where RTT becomes between RTTmin and the RTTmin plus maximum queuing delay (about 12 ms). On the other hand, the queue size in RTTmin = 20, 50, 100, and 200 ms is set to the BDP of setting RTTmin, where RTT becomes between RTTmin and twice the RTTmin. To create network congestion, a TCP sender generates short-lived TCP flows (TCP-Sack) using a Poisson process with an average rate of 30 flows per second, and the size of a TCP flow follows a Pareto distribution with an average of 40 packets and a shape parameter of 1.5. We used a single TFRC streaming flow with a maximum data rate of 30 Mbps, or a single normal UDP streaming flow transmitting at a maximum data rate of 30 Mbps. Each simulation ran for about 120 sec.

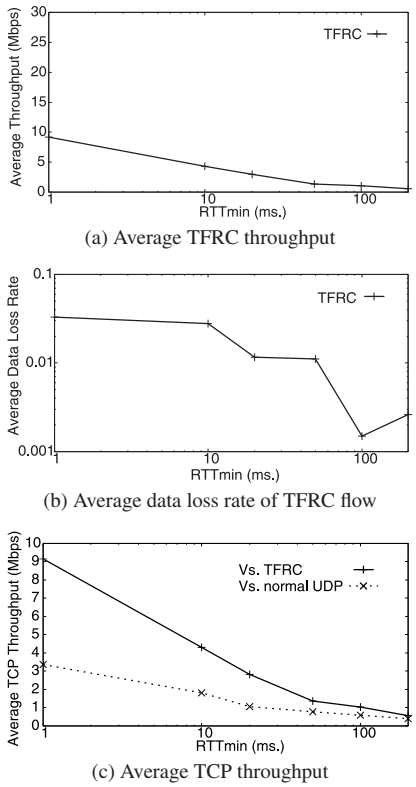


Fig. 2 The average throughput of TFRC flow, the average data loss rate of TFRC flow, and the average throughput of TCP flows competing a TFRC or normal UDP flow under varied minimum RTTs.

Figure 2 (a) shows that because the expected throughput of a competing TCP flow that the TFRC flow calculates becomes small by occurrences of packet loss caused by TCP flows, a TFRC flow cannot maintain the maximum data transmission rate under all minimum RTTs. Also, since the expected TCP throughput becomes small as RTT increases, the average throughput of the TFRC flow decreases with an increase in RTTmin (i.e., BDP). As shown in Fig. 2 (b), since an increasing congestion window size during TCP slow-start or congestion avoidance phase causes bursty packet losses, the average data loss rate of a TFRC flow – even with minimum RTTs more than 50 ms – becomes more than 0.1% in addition to the reduction in the transmission rate. Figure 2 (c) shows the average throughput of TCP flows competing with a normal UDP or TFRC flow. Since TCP is inefficient in high BDP links, with minimum RTTs of more than 50 ms, a normal UDP flow does not have a negative impact on the average throughput of TCP flows. This means that a TFRC flow did not effectively consume network resources by unnecessary reductions in the data transmission rate (as shown in Fig. 2 (a)). In this context, a TFRC flow, which competes for bandwidth with TCP flows in higher BDP environments, tends to suffer from a degradation of streaming quality caused by packet losses that TCP flows induce.

2.3 Requirement

As described in Section 2.1, high-performance real-time streaming using TFRC suffers from bursty packet losses that competing TCP flows induce, and unnecessary reductions in the data

transmission rate (i.e., video quality) due to improper congestion control. Existing congestion control protocols thus overemphasize the maintenance of TCP-friendliness too much, which leads to low network utilization especially in high BDP links. Instead of using TFRC, an alternative protocol is needed for high-performance real-time streaming to avoid a degradation of playback quality caused by packet losses and to preserve the highest data transmission rate in congested high speed networks where the bottleneck link bandwidth is much larger than the total consumption bandwidth of high-performance streaming flows. From this viewpoint, it is important to protect playback quality from packet loss while executing effective congestion control to better utilize network resources without interacting badly with existing TCP traffic.

Forward Error Correction (FEC) is the well-known algorithm that has been notably used in streaming applications to improve playback quality in the presence of data packet losses. An application level forward error correction (AL-FEC) approach prevents retransmission delays by preventively adding redundant packets to the streaming data flow. Thanks to the redundancy, a certain number of missing data packets can be recovered. However, ascertaining and/or controlling optimal FEC redundancy in congested networks 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 future packet loss pattern, 2) if all streaming flows constantly add a large amount of redundant packets to recover busty packet losses, bandwidth overhead becomes large and has an adverse effect on competing flows, and 3) increasing FEC redundancy may disturb competing flows when the conditions of competing flows are sensitively oscillated in the network. Block interleaving techniques are frequently used to improve FEC performance by reducing the burstiness of packet losses. However, since interleaving method involves buffering and computation delays, in this study we do not use it to keep delays to a minimum. The following points thus must be considered when enhancing FEC redundancy in congested networks:

- TCP flows tend to transmit their packets in bursts especially in high BDP links, and therefore may cause bursty packet losses [30]. Because this situation often makes it difficult to appropriately adjust FEC redundancy (such that FEC cannot recover lost data packets), an occurrence of bursty packet loss should be suppressed as much as possible 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 adjust the degree of FEC redundancy in order to minimize the adverse effect on performance of competing flows as much as possible.

3. Streaming Control Algorithm Using GMIAD

In this section, we describe GENEVA, the streaming control algorithm using generalized multiplicative-increase/additive-decrease (GMIAD) that satisfies the aforementioned requirements, and analyze the parameter settings.

3.1 Design Overview

GENEVA is designed as an end-to-end model using the following criteria:

(1) To achieve high network utilization, GENEVA allows a high-performance streaming flow to maintain moderate network congestion, in which packet loss rate is less than $p_{\max}$ (a predefined constant value).

(2) A sender with GENEVA needs to transmit data packets at an almost constant interval time, and control the degree of FEC redundancy to its data stream in congested networks. This situation may adversely increase traffic congestion. An GENEVA flow thus should converge at an appropriate equilibrium point to recover data packet losses under stable conditions during transmission. To achieve such a condition, supporting window control is needed to specify an acceptable degree of FEC redundancy.

(3) To suppress an occurrence of bursty packet loss that TCP flows cause and improve FEC recover capabilities, the degree of FEC redundancy is increased to prevent their congestion window sizes from increasing considerably. At the same time, the increase in FEC redundancy should not interact badly with competing TCP flows.

In accordance with the aforementioned design overview (3), to improve high-performance streaming quality by FEC recovery of lost data packets, GENEVA should try to keep or increase the degree of FEC redundancy even in the presence of packet loss while utilizing expected available network resources which TCP flows cannot consume by their nature, and prevent TCP flows from achieving larger window size so as not to pose bursty packet loss. From this point of view, GENEVA does not adopt the well-deployed Additive-Increase/Multiplicative-Decrease (AIMD) algorithm [34] that decreases the transmission rate reacting to packet loss, but adopts GMIAD that increases the transmission rate reacting to packet loss (i.e., GMIAD increases the degree of FEC redundancy in GENEVA). Meanwhile, GENEVA using GMIAD decreases the degree of FEC redundancy during the period of no packet loss, where unnecessary FEC redundancy is suppressed so as not to trigger further network congestion in accordance with the design overview (2).

Figure 3 illustrates the FEC overview. GENEVA operates

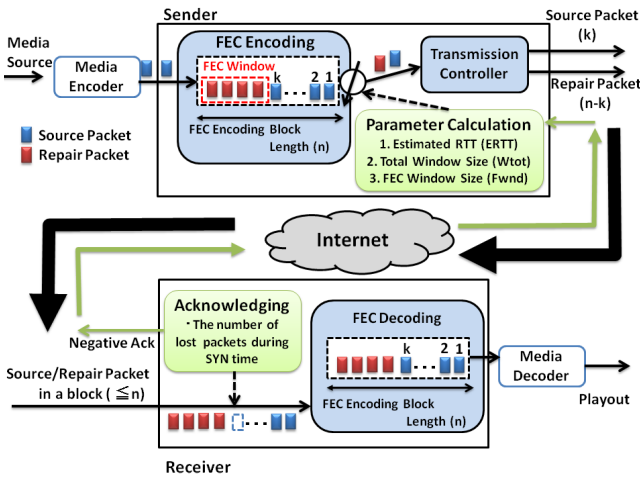


Fig. 3 GENEVA overview.

mainly at the sender end. A sender transmits data and FEC repair packets over a real-time transport protocol (RTP) [6] carried on top of the Internet Protocol (IP) and UDP. The GENEVA sender collects the feedback information transmitted over a real-time transport control protocol (RTCP) delivered by a receiver, then adjusts the degree of FEC redundancy based on packet loss conditions and estimated RTT. The data transmission rate, which depends on the video format preliminarily assigned, is maintained during transmission.

GENEVA leverages an AL-FEC, in which $n - k$ FEC repair packets are added to a block of k data packets (FEC source packets). We consider maximum distance separable (MDS) codes such as Reed-Solomon codes [7] which can recover all of the missing packets from any set of exactly k packets. Here, we define the number of FEC repair packets as the "FEC window size $Fwnd$ (packets)," 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. To calculate and create FEC repair packets in each block, all of the generated data packets (FEC source packets) are stored once in the FEC encoding block buffer. Depending on the FEC window size, the number of FEC repair packets stored in the FEC encoding block buffer varies.

3.2 Algorithm and Analysis

We now describe the GENEVA algorithm and its parameter settings based on the aforementioned design criteria.

3.2.1 Acknowledging

To control network congestion by adjusting the $Fwnd$, an GENEVA sender needs negative acknowledgements of packet reception from a corresponding receiver. Since GENEVA tolerates moderate network congestion without reductions in the data transmission rate (unlike TCP), generating negative acknowledgements may consume large bandwidth. To reduce the ratio of bandwidth consumed by control traffic, an GENEVA receiver sends feedback information at constant time interval (SYN), which denotes the number of lost packets during SYN time. The SYN is related to responsiveness or stability of GENEVA flow (i.e., conditions of FEC recovery capabilities)[*1]. GENEVA thus maintains the FEC window size greater than or equal to the minimum FEC window size ($Fwnd_{\min}$) to allow the flow to recover lost source packets in moderate network congestion[*2]. In GENEVA, SYN time is currently set to 0.01 sec based on our comprehensive simulation results.

3.2.2 FEC Source and Repair Transmission

Figure 4 shows the transmission of FEC source and repair packets. GENEVA adjusts the $Fwnd$ in an arbitrarily fixed FEC source block length (k) during transmission. The value of k is chosen to be the number of FEC source packets needed for a sender to transmit during SYN time (0.01 sec), which depends on the video format and its encoding parameters. For instance, on the assumption that a sender transmits RTP/HDV (MPEG2-TS) packets of 1,500 bytes at a rate of 25 Mbps [10], the value of k is calculated as follows: $k = 25\,(\text{Mbps}) * 10^6 * 0.01\,(\text{sec})/(1500 * 8) = 20.83$. Thus, in the case of transmission RTP/HDV, the value of k becomes 20 or 21. The generated FEC repair packets for each source block are sent together with the corresponding source packets within SYN time. Thus, each time a sender receives feedback information, $Fwnd$ is adjusted using the increase/decrease algorithm (described below). It is well understood from queuing theory that burst transmission of packets poses bursty packet loss that limits FEC recovery performance [39], [40]. To avoid a generation of bursty traffic, the transmission controller evenly spaces FEC source and repair packets sent into the network over SYN time by controlling the inter-packet gap (IPG). The IPG time during an arbitrary SYN time becomes as follows: $IPG = \text{SYN}/(Fwnd + k)$.

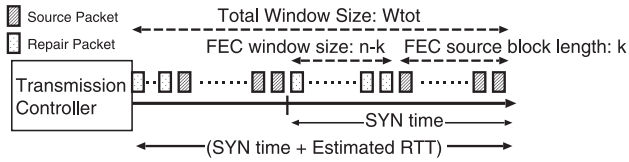


Fig. 4 FEC source and repair transmission.

Using feedback information, a sender calculates the sample RTT ($SRTT$) to derive the number of unacknowledged packets sent to the network, which is denoted by the "current total window size (W_{tot})." The W_{tot} is used to adjust the $Fwnd$ with the increase/decrease algorithm. Keeping pairs of the sequence number and its transmitted time, a sender calculates $SRTT$. Using an exponential weighted moving average with the smoothing factor of α ($0 < \alpha < 1$), the estimated RTT ($ERTT$) is calculated as follows: $ERTT = \alpha * ERTT + (1 - \alpha) * SRTT$. Since the W_{tot} denotes the number of unacknowledged packets sent to the network (i.e., the number of packets sent to the network during $ERTT$ plus SYN time), the W_{tot} is dynamically derived using $ERTT$, SYN time and the current IPG, as follows:

$$\begin{aligned} W_{tot} &= (ERTT + \text{SYN})/IPG \\ &= (Fwnd + k) * (ERTT + \text{SYN})/\text{SYN} \end{aligned} \tag{1}$$

3.2.3 FEC Window Size Adjustment

We define a controller that forces the total window size to achieve an equilibrium point (W^*_{tot}) where the $Fwnd$ stably converges at a packet loss probability $p < p_{\max}$:

$$W^*_{tot} = a/(p_{\max} - p) \tag{2}$$

where a is a constant with $a > 0$, and is the scaling property of W_{tot}. This equation shows that, as the network becomes more congested (i.e., p increases), W^*_{tot} also becomes larger, which results in an increase in the $Fwnd$. Since we consider a high-performance real-time application that consumes at least 25 Mbps (W_{tot} becomes about 42 under 10 ms RTT), we set a to 2 to scale up the $Fwnd$. Using Eq. (2), the $Fwnd$ in the steady state is shown as follows:

*1 Note that since GENEVA stores the k source packets in each block to generate the $Fwnd$ repair packets, it takes a certain time for a sender to adjust and send $Fwnd$ repair packets after the last adjustment.

*2 In a situation in which feedback information is lost, GENEVA performance (i.e., FEC recovery capabilities) is not sensitively affected due to the preliminarily operated $Fwnd$. In addition, GENEVA maintains the current $Fwnd$ without increasing the $Fwnd$, and therefore does not induce an adverse impact on performance of other competing flows.

$$Fwnd = \lfloor (W^*_{tot} * \mathrm{SYN}/(ERTT + \mathrm{SYN}) - k \rfloor \quad (3)$$

When the $Fwnd < Fwnd_{\min}$, GENEVA maintains the $Fwnd$ at $Fwnd_{\min}$. Note that although a larger W_{tot} that a flow has from the first (e.g., with high throughput and/or longer RTT) cannot approach the W^*_{tot}, the $Fwnd$ converges at small sizes, which contributes to stability of other competing GENEVA flows.

GENEVA adopts the Generalized Multiplicative Increase and Additive Decrease (GMIAD) algorithm to achieve an equilibrium point in Eq. (3). When a sender receives feedback information indicating the number of both lost and delivered packets during SYN time, the following calculation is repeated for a number of times equal to the number of lost packets, to first update the W_{tot}:

$$W_{tot} \leftarrow W_{tot} + i(W_{tot}) \quad (4)$$

where $i(W_{tot})$ is an incremental function of W_{tot}. Then, the W_{tot} is derived by repeating the following calculation for a number of times equal to successfully delivered packets:

$$W_{tot} \leftarrow W_{tot} - b \quad (5)$$

where b is a constant with $b > 0$.

The behavior of the aforementioned GMIAD algorithm is described with a differential equation:

$$\frac{dW_{tot}}{dt} = i(W_{tot})\frac{W_{tot}}{(\mathrm{RTT} + \mathrm{SYN})}p - b\frac{W_{tot}}{(\mathrm{RTT} + \mathrm{SYN})} \quad (6)$$

At an equilibrium point, the following equation is derived by Eq. (6):

$$i(W^*_{tot}) = \frac{b}{p} \quad (7)$$

Using Eqs. (2) and (7), $i(W_{tot})$ is given with the constant value $p_{\max}$ and b:

$$i(W_{tot}) = bW_{tot}/(p_{\max}W_{tot} - 2) \quad (8)$$

3.2.4 Parameter Settings

The value of α for the estimated RTT ($ERTT$) calculation determines how rapidly $ERTT$ adapts to the sample RTT ($SRTT$) changes. $ERTT$ using a lower value of α adapts to $SRTT$ changes more rapidly. In a situation in which $SRTT$ fluctuates significantly, $ERTT$ using a lower value of α also fluctuates significantly. This situation produces large fluctuations in W_{tot}, and the fluctuated W_{tot} determines the $Fwnd$ irrespective of packet loss conditions. Since as shown in Eq. (2), the equilibrium point of W_{tot} is defined by using packet loss probability (p), such a situation may cause improper adjustments of the $Fwnd$. To avoid it, α is set to relatively high value, 0.9.

The setting of $p_{\max}$ defines the upper limit of packet loss probability that an GENEVA flow can tolerate. Because GENEVA with a high value of $p_{\max}$ increases the Fwnd up to a maximum extent under high loss rate regimes (i.e., packet loss probability is around the $p_{\max}$), the value of $p_{\max}$ relates to friendliness for competing TCP flows and GENEVA aggressiveness. Since packet loss plays a key role in achievable TCP performance [32], [38], GENEVA with high value of $p_{\max}$ thus may cause an adverse impact on competing TCP performance or congestion collapse. Conventional wisdom holds that a loss rate of more than 5% has a significant adverse effect on TCP performance, because it will greatly limit the size of the congestion window and hence the transfer rate, while 3% is often substantially less serious [3]. Thus, $p_{\max}$ of upper limit of packet loss probability that GENEVA flow can tolerate is set to 0.03 in this work.

When $Fwnd_{\min}$ is set to $\lceil k * p_{\max} \rceil$ in association with $p_{\max}$, GENEVA with the k of about 20 keeps the $Fwnd$ of 1 in the absence of packet loss and cannot recover lost data packets by sudden bursty packet loss. Therefore, $Fwnd_{\min}$ is set to a constant value of 8 in this work. Also, it is important to decide the maximum acceptable $Fwnd$, because the additional delay by FEC encoding/decoding becomes large with the increase in 1) k and the $Fwnd$ (i.e., encoding time) and 2) packet loss rate (i.e., the decoding time becomes large with the increase in the number of recovered packets within a block). According to the work using a real implementation with Reed-Solomon codes for a high-performance real-time application transmitting at a rate of 30 Mbps [10], the additional video frame delay compared to the case without FEC becomes lower about 150 ms within less than 15% of packet loss rate, where the value of k and the number of repair packets within a block were set to 170 and 85 respectively. On the assumption that the value of k that GENEVA holds becomes smaller than 170, the maximum acceptable $Fwnd$ of GENEVA is set to 60 so as to suppress the additional delay to below 150 ms. GENEVA thus requires the equivalent buffer length of about 150 ms at the cost of the real-time performance. If the packet loss probability continues to be more than or equal to $p_{\max}$, GENEVA keeps the maximum acceptable $Fwnd$ in the presence of the high packet loss rate, which results in the significant degradation of competing TCP performance and congestion collapse. GENEVA thus needs to reduce the data transmission rate at the cost of video quality to avoid congestion collapse. In this study, we do not assume such a situation where the physical bandwidth is notably less than the averaged consumption bandwidth of TCP and high-performance streaming flows.

Figure 5 shows the GENEVA scaling properties of W_{tot} when the packet loss probability $p = p1, p2$ ($0 < p1 < p2 < p_{\max}$). Whereas W_{tot} increases by $i(W_{tot})$ in response to packet loss, W_{tot} decreases based on Eq. (5) during periods of no packet loss. Although the W_{tot} in $p2$ (W_{tot}_2) becomes larger than the W_{tot} in $p1$ (W_{tot}_1), both the increase ratio $i(W_{tot}\_2)$ and decrease speed up

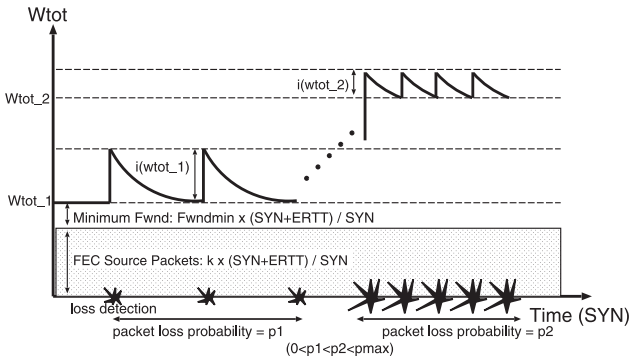


Fig. 5 GENEVA scaling properties.

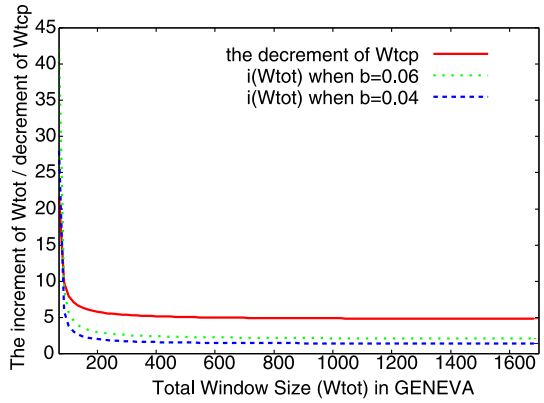


Fig. 6 The increment of W_{tot} ($i(W_{tot})$) and decrement of TCP window size (W_{tcp}) as a function of W_{tot}. From the aspect of design that $i(W_{tot})$ should not exceeds the corresponding decrement of $Wtcp$, the constant b (decrement function in GMIAD algorithm) is set to 0.04.

to W_{tot}_2 become smaller compared those in $p1$.

The increase in $Fwnd$ based on $i(W_{tot})$ should be adjusted properly to minimize the adverse effect on competing TCP performance by utilizing network resources that TCP fails to copy. In TCP, the window size (W_{tcp}) is halved on detecting a packet loss during a round trip time, and becomes $W_{tcp}/2$. The constant value b, which relates to $i(W_{tot})$ as shown in Eq. (8), should be set to prevent $i(W_{tot})$ from exceeding the decrease in TCP window size (namely, $W_{tcp}/2$), in order to achieve stable conditions. **Figure 6** shows both $i(W_{tot})$ and $W_{tcp}/2$ as a function of W_{tot}, by using the TCP performance model [32]. The range of W_{tot} is set to between 70 and 1,700. The W_{tot} value of 1,700 represents about the size of W_{tot} that GENEVA at a rate of 25 Mbps achieves with the maximum acceptable $Fwnd$ (i.e., $Fwnd = 60$) under RTT 200 ms: $W_{tot} = (60+21)*(0.2+0.01)/0.01 = 1701$, as shown in Eq. (1). When W_{tot} of larger value than $2/p_{\max}$ approaches $2/p_{\max}$ (i.e., about 66.6), $i(W_{tot})$ approaches positive infinity (as shown in Eq. (8)). The range of W_{tot} thus starts from 70. When b becomes large, GENEVA becomes aggressive to increase the $Fwnd$ with an increase in the W_{tot}. As you can see in Fig. 6, each $i(W_{tot})$ with b = 0.06 and 0.04 does not exceed $W_{tcp}/2$ at an almost full range of W_{tot}. However, each $i(W_{tot})$ with b = 0.06 and 0.04 exceeds the $W_{tcp}/2$ in below about 80 and 73 of W_{tot}, respectively. In this work, we set b to 0.04. In below 73 of W_{tot}, $i(W_{tot})$ is constantly set to 10 so as not to exceed $W_{tcp}/2$. Note that although $i(W_{tot})$ in large W_{tot} becomes small, GENEVA tends to gradually increase the $Fwnd$ in conditions of bursty packet loss by reacting to every packet loss event.

4. Evaluation

4.1 Simulation Setup and Performance Metrics

Using an NS-2 extended with GENEVA module, we performed experiments using the dumbbell topology shown in **Fig. 7**. The bandwidth of the bottleneck link was set to 1 Gbps. Each sender and receiver were connected to the bottleneck link through the 10 Gbps access link with propagation delay of about 1 ms. The two-way propagation delay (i.e., minimum RTT, RTTmin) was set to 10 or 100 ms. According to RTTmin settings, the propagation delay in the bottleneck link varies, and BDP is calculated using the RTTmin. The packet size was set to 1,500 bytes for all

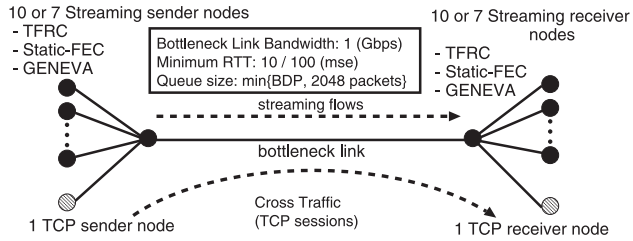


Fig. 7 The simulation topology (single bottleneck link).

connections. A drop-tail queuing was used at the router in the bottleneck link, the queue length size was set to min{BDP, 2048 (packets)}. The queue length size in RTTmin of 10 ms was set to BDP of setting RTTmin 10 ms, where RTT becomes the RTTmin and twice the RTTmin. In RTTmin of 100 ms, BDP in packets is calculate as follows: $BDP = 1\,(\text{Gbps}) * 10^9 * 0.1\,(\text{sec})/(1500*8) = 8333.33$. Since the value of the queue size in packets exceeds 2,048, the queue size in RTTmin of 100 ms was set to 2,048, where RTT becomes between the RTTmin and the RTTmin plus maximum queuing delay (about 25 ms). About 25% of the BDP is available as buffers on the bottleneck link when the RTTmin is 100 ms; A decrease in available queuing delay is favorable in terms of the cost of implementing high-speed memory systems in network devices, and also for end system applications. Each simulation ran for about 3 minutes.

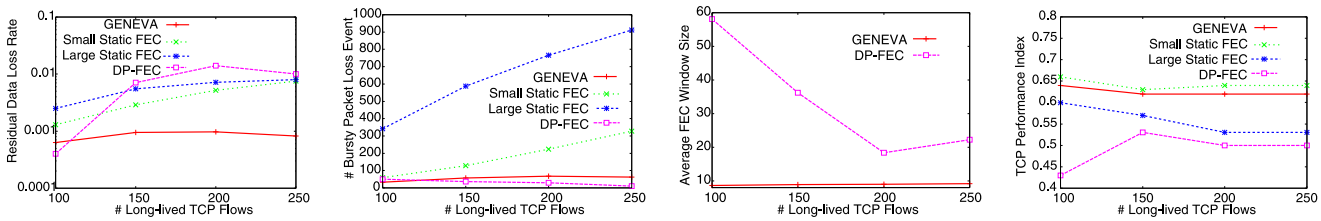
As the GENEVA performance metrics, we observed 1) the average residual data loss rate of GENEVA flows which denotes the ratio of the total number of non-recovered data packets to the total number of sent data packets, 2) the average number of "bursty packet loss events," and 3) the average throughput of competing TCP flows. We here define the bursty packet loss event as the situation in which more than three packets are consecutively lost in an GENEVA flow, and expect that GENEVA suppresses the number of occurrence of bursty packet loss event to improve FEC recovery capabilities.

The metrics of the average residual data loss rate and the average number of bursty packet loss events were compared with those observed when we used small/large Static-FEC flows (i.e., the $Fwnd$ is constant) and DP-FEC flows [29] under the same network condition. We set the $Fwnd$ of small and large Static-FEC flow to $Fwnd_{\min}$ and 1.5 times the $Fwnd_{\min}$, respectively. As TCP Performance index ($TPindex$), we use the result of the average throughput of TCP flows observed when they compete with TFRC flows, and define $TPindex$ as follows:.

$$TPindex = \frac{TCPthuput\_Target}{TCPthuput\_TCPstr} \tag{9}$$

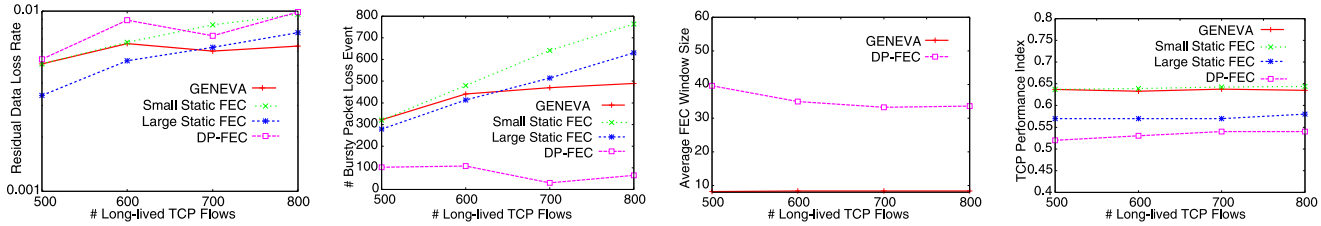
where $TCPthuput\_Target$ is the result of the average throughput of TCP flows competing with small/large Static-FEC or DP-FEC or GENEVA flows; $TCPthuput\_TCPstr$ is the result of the average throughput of TCP flows competing with TFRC flows under the same network condition.

We used two types of TCP flows using TCP-SACK, 1) short-lived TCP flows and 2) long-lived TCP flows. Short-lived TCP 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 Pareto distribution with an average of s_tcp packets and shape parameter 1.5. We define the load of TCP flows as



(a) Residual Data Loss Rate (RTTmin 10 ms) (b) The number of bursty packet loss event (RTTmin 10 ms) (c) Average FEC window size (RTTmin 10 ms) (d) TCP performance index (RTTmin 10 ms)

Fig. 8 The results of the performance of 10 GENEVA flows under RTTmin 10 ms in competition with long-lived TCP flows. Three metrics (residual data loss rate, the number of bursty packet loss events and TCP performance index) were compared to those of 10 small/large Static-FEC flows and 10 DP-FEC flows under the same network condition.



(a) Residual Data Loss Rate (RTTmin 100 ms) (b) The number of bursty packet loss event (RTTmin 100 ms) (c) TCP performance index (RTTmin 100 ms) (d) TCP performance index (RTTmin 100 ms)]

Fig. 9 The results of the performance of 10 GENEVA flows under RTTmin 100 ms in competition with long-lived TCP flows. Three metrics (residual data loss rate, the number of bursty packet loss events and TCP performance index) were compared to those of 10 small/large Static-FEC flows and 10 DP-FEC flows under the same network condition.

$\rho\_tcp = r\_tcp * s\_tcp$. Long-lived TCP flows are persistent in the network.

4.2 Homogeneous GENEVA Flows VS. TCP Flows

In this experiment, all GENEVA or small/large Static-FEC or DP-FEC flows transmit data packets at a rate of 30 Mbps. We set the number of streaming flows to 10, and evaluated the GENEVA performance in competition with long-lived TCP flows or short-lived TCP flows in network conditions where the total packet loss rate on the bottleneck link varies from about 1% to 4%. This range of the packet loss ratio is based on the 2012 yearly reports of the ICFA-SCIC Monitoring WG [3] indicating that the packet loss rate on the Internet is around 1% on average.

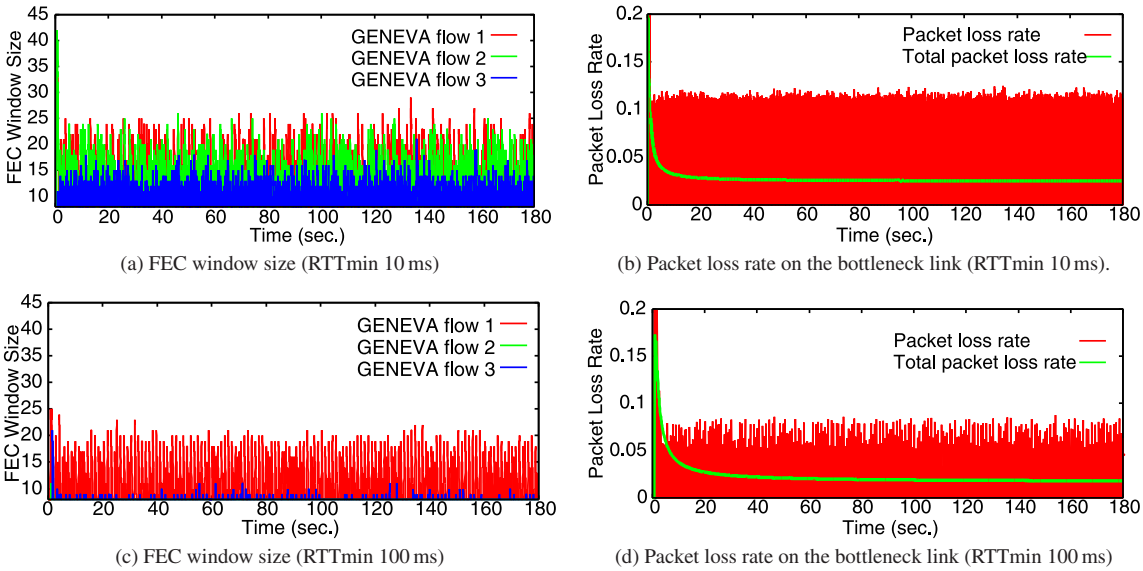
4.2.1 Competition with Long-lived TCP Flows

Figures 8 and **9** shows the results of the experiments where 10 GENEVA flows or 10 small/large Static-FEC flows or 10 DP-FEC flows compete with long-lived TCP flows under the RTTmin 10 ms and 100 ms, respectively. Under both of the RTTmin (10 and 100 ms), as the number of competing TCP flows increases, the number of bursty packet loss events of the GENEVA flows becomes lower than that of the small Static-FEC flows. Because of this, the corresponding residual data loss rate also decreases. The $TPindex$ of both small Static-FEC and GENEVA flows becomes about 0.65, because they preserve the highest data transmission rate (i.e., 30 Mbps) unlike TFRC. Although the GENEVA flows increase their $Fwnd$ in response to packet losses, the $TPindex$ is almost the same as that of the small Static-FEC flows (about 0.65). This means that the GENEVA flows effectively utilized network resources that TCP flows fail to copy, by properly adjusting the $Fwnd$.

As shown in Fig. 8 (b), the large Static-FEC flows under the RTTmin 10 ms cause more bursty packet loss events than that of small Static FEC flows or GENEVA flows, which results in higher residual data loss rate of the large Static-FEC flows. In addition, as seen from Fig. 8 (d), the corresponding $TPindex$ of the large static-FEC flows becomes lower than that of GENEVA flows due to the constantly added large FEC redundancy. In the case of the RTTmin 100 ms, since TCP flows become less aggressive compared to the case of the RTTmin 10 ms, the number of bursty packet loss events of the large Static-FEC flows becomes lower than that of the small Static FEC flows, and the corresponding residual data loss rate also becomes lower (as seen from Fig. 9 (b) and 9 (a)). Although the residual data loss rate of the large Static-FEC flows competing with 500 or 600 TCP flows becomes lower than that of the GENEVA flows, the $TPindex$ of the large Static-FEC flows becomes lower than that of GENEVA flows (as seen from Fig. 9 (d)).

Since DP-FEC probes the available bandwidth by increasing the $Fwnd$, the DP-FEC flows under the RTTmin 10 ms and 100 ms maintain a much larger Fwnd, compared to that of the GENEVA flows (as seen from Figs. 8 (c) and 9 (c)). However, since their increased $Fwnd$ induces residual data packet loss as a side effect between the competing DP-FEC flows, the residual data loss rate becomes higher than that of the GENEVA flows, except in the case where the DP-FEC flows under the RTTmin 10 ms compete with 100 TCP flows (as seen from Figs. 8 (a) and 9 (a)). Moreover, due to the large average $Fwnd$, the $TPindex$ of the DP-FEC flows becomes lower than that of large Static-FEC flows.

Whereas the average $Fwnd$ of the GENEVA flows under the RTTmin 10 ms increases up to about 9.2 (in competition with 250 TCP flows), the $Fwnd$ under the RTTmin 100 ms maintains almost the same value (between 8.2 and 8.4). This is because GENEVA flows under the longer RTT have larger total window size. This behavior contributes to the stability. **Figure 10** shows the trace of the $Fwnd$ of the selected 3 GENEVA flows, and the



(a) FEC window size (RTTmin 10 ms)

(b) Packet loss rate on the bottleneck link (RTTmin 10 ms).

(c) FEC window size (RTTmin 100 ms)

(d) Packet loss rate on the bottleneck link (RTTmin 100 ms)

Fig. 10 Trace of FEC window size of three selected GENEVA flows and packet loss rate on the bottleneck link. 10 GENEVA flows under RTTmin 10 and 100 ms compete with 200 and 700 long-lived TCP flows, respectively.

Table 1 The results of 10 GENEVA flows, 10 small/large Static-FEC flows and 10 DP-FEC flows under RTTmin 10 and 100 ms in competition with short-lived TCP flows.

TCP load(ρ_tcp): 500 Mbps	Data Rate (Mbps)	Data Loss Rate	Packet Loss Rate	FEC Window Size	Total Window Size	# Bursty Packet Loss Event	TCP performance index	Link Loss Rate
RTTmin: 10 ms, r_tcp = 12.5	avg.	avg.	avg.	avg.	avg.	avg.	avg.	avg.
10 GENEVA flows	30.0	0.0011	0.0068	10.19	105.26	164.90	0.57	0.039
10 Small Static-FEC flows	30.0	0.0029	0.0078	8.0	—	305.70	0.61	0.040
10 Large Static-FEC flows	30.0	0.0026	0.0083	12.0	—	281	0.49	0.039
10 DP-FEC flows	30.0	0.0024	0.32	48.12	—	169.7	0.10	0.024
RTTmin: 10 ms, r_tcp = 25								
10 GENEVA flows	30.0	0.0013	0.0070	9.88	110.56	283.60	0.61	0.033
10 Small Static-FEC flows	30.0	0.0029	0.0089	8.0	—	509.40	0.65	0.035
10 Large Static-FEC flows	30.0	0.0032	0.0093	12.0	—	938.5	0.46	0.037
10 DP-FEC flows	30.0	0.0061	0.23	35.69	—	610	0.18	0.025
RTTmin: 100 ms, r_tcp = 12.5								
10 GENEVA flows	30.0	0.0039	0.0074	8.19	424.34	393.50	0.77	0.030
10 Small Static-FEC flows	30.0	0.0044	0.0077	8.0	—	452.0	0.78	0.030
10 Large Static-FEC flows	30.0	0.0050	0.0093	12.0	—	549.5	0.63	0.031
10 DP-FEC flows	30.0	0.0016	0.35	58.11	—	878.6	0.22	0.025
RTTmin: 100 ms, r_tcp = 25								
10 GENEVA flows	30.0	0.0036	0.0069	8.14	425.41	459.60	0.82	0.023
10 Small Static-FEC flows	30.0	0.0037	0.0070	8.0	—	473.20	0.82	0.024
10 Large Static-FEC flows	30.0	0.0036	0.0079	12.0	—	688.6	0.70	0.027
10 DP-FEC flows	30.0	0.0021	0.25	40.41	—	1186.0	0.33	0.030

packet loss rate in the bottleneck link averaged over RTTmin interval and also its total packet loss rate. Because each of packet loss rates that the GENEVA flows observed is different, the variation of the corresponding $Fwnd$ is also different. However, under both of the RTTmin, the GENEVA flows keep both of the $Fwnd$ of each flow and the link loss rate stable.

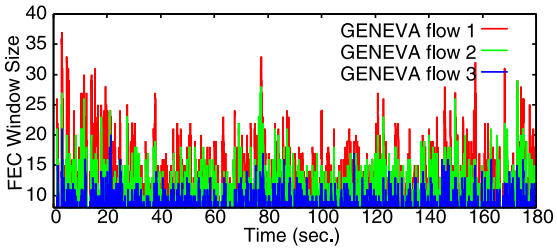
4.2.2 Competition with Short-lived TCP Flows

To investigate the effect of TCP slow-start on the GENEVA performance, we set the experiments where 10 streaming flows compete with short-lived TCP flows. The load of short-lived TCP flows (ρ_tcp) was set to 50% of the bottleneck link capacity, and the rate of an occurrence of TCP flow per second (r_tcp) was set to 12.5 or 25. **Table 1** shows the results of the experiments. Under the RTTmin 10 ms, the GENEVA flows suppress the number of bursty packet loss events, compared to that of small/large Static-FEC flows. The suppression contributes to the reduction in the residual data loss rate. The $TPindex$ of the GENEVA flows becomes slightly lower than that of the small Static-FEC flows and becomes higher than that of the large Static-FEC flows, because the added FEC redundancy effectively prevents the congestion window size of TCP flows from increasing aggressively. Since the DP-FEC flows under the RTTmin 10 ms has the large average $Fwnd$, the $TPindex$ decreases by more than 0.4 compared to that of the GENEVA flows in both of r_tcp = 12.5 and 25. In addition, the residual data loss rate of the DP-FEC flows becomes higher than that of the GENEVA flows.

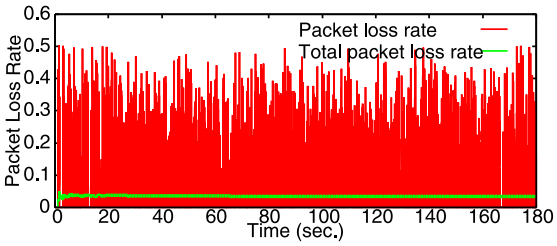
On the other hand, under the RTTmin 100 ms, the data loss rate of the GENEVA flows was not significantly improved especially in r_tcp = 25, due to the large total window size caused by the longer RTTmin 100 ms. However, the GENEVA flows reduce the number of bursty packet loss events, and maintain almost the same $TPindex$ as the small Static-FEC flows and higher $TPindex$

Table 2 The results of 7 GENEVA flows, 7 small/large Static-FEC flows and 7 DP-FEC flows with different data transmission rates under RTTmin 10 ms, in competition with 200 long-lived TCP flows.

	Data Rate (Mbps)	Data Loss Rate	Packet Loss Rate	FEC Window Size	Total Window Size	# Bursty Packet Loss Event	TCP performance index	Link Loss Rate
RTTmin: 10 ms # TCP flows:200	avg.	avg.	avg.	avg.	avg.	avg.	avg.	avg.
5 GENEVA flows	30.0	0.0014	0.012	9.28	143.75	110.6	—	—
1 GENEVA flow	60.0	0	0.0095	9.87	273.35	29	—	—
1 GENEVA flow	90.0	0	0.011	9.21	413.25	4	—	—
Total (avg.)	30.0	0.0010	0.011	9.35	200.77	83.71	0.62	0.026
5 Small Static-FEC flows	30.0	0.0027	0.020	8.0	—	416.4	—	—
1 Small Static-FEC flow	60.0	0	0.0089	8.0	—	13	—	—
1 Small Static-FEC flow	90.0	0	0.011	8.0	—	0	—	—
Total (avg.)	42.9	0.0039	0.016	8.0	—	299.29	0.63	0.027
5 Large Static-FEC flows	30.0	0.0072	0.023	12.0	—	873	—	—
1 Large Static-FEC flow	60.0	0	0.0080	12.0	—	2	—	—
1 Large Static-FEC flow	90.0	0	0.0096	12.0	—	2	—	—
Total (avg.)	42.9	0.0051	0.019	12.0	—	624.14	0.60	0.027
5 DP-FEC flows	30.0	0	0.00013	52.1	—	11	—	—
1 DP-FEC flow	60.0	1.1e-06	0.00084	40.9	—	22	—	—
1 DP-FEC flow	90.0	0.032	0.037	7.2	—	23	—	—
Total (avg.)	42.9	0.0046	0.0055	44.09	—	14.29	0.49	0.030



(a) FEC window size



(b) Packet loss rate on the bottleneck link

Fig. 11 Trace of FEC window size of three selected GENEVA flows competing with short-lived TCP flows under RTTmin 10 ms, and packet loss rate on the bottleneck link. In this experiment, the load of TCP flows (ρ_tcp) was set to 50% of the bottleneck link capacity, and the rate of an occurrence of TCP flow per second (r_tcp) was set to 25.

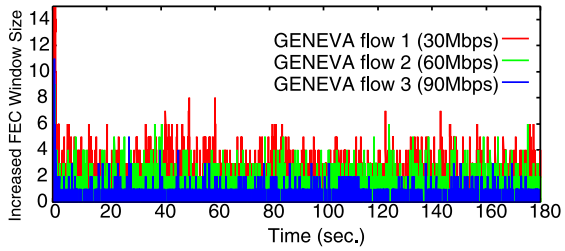
than that of the large Static-FEC flows. Although the residual data loss rate of the DP-FEC flows in both of r_tcp = 12.5 and 25 under the RTTmin 100 ms becomes lower than that of the GENEVA flows, the $TPindex$ of the DP-FEC flows decreases considerably by more than 0.49. Thus, the DP-FEC flows fail to prevent adverses effect on TCP performance. However, the residual data loss rate of the GENEVA flows becomes less than or equal to small/large Static FEC flows, and the GENEVA flows maintain almost the same $TPindex$ as the small Static-FEC flows by having the function of the total window size control to adjust the $Fwnd$.

Figure 11 shows the trace of one of the results in the experiments. We can see that, although the packet loss rate fluctuates and bursty packet losses occur, both of the $Fwnd$ of the GENEVA flows and the total packet loss rate on the bottleneck link become stable.

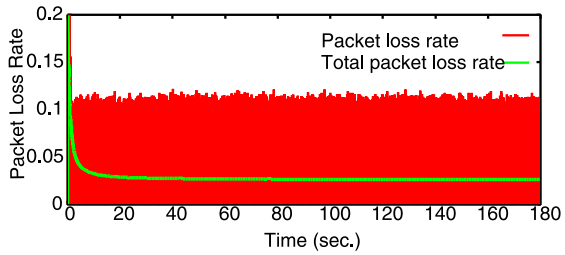
Note that since 1) the number of bursty packet loss events and residual data loss rate of the GENEVA flows became less than that of the small Static-FEC flows and 2) the $TPindex$ of GENEVA flows became almost the same as that of the small Static-FEC flows, the increased $Fwnd$ did not induce more residual data packet loss and degradation of $TPindex$ as a side effect for the competing flows. However, when there is not fully available bandwidth in a congested link (e.g., in case that the bottleneck link bandwidth is much less than 1 Gbps), the increased $Fwnd$ induces packet loss up to more than $p_{\max}$, where the residual data loss rate increases and the $TPindex$ degrades. In such condition, it would be necessary to reduce the data transmission rate and degrade the video quality, which we have not assumed as described in Section 1.

4.3 Heterogeneous GENEVA Flows VS. TCP Flows

To confirm the stability of heterogeneous GENEVA flows, we performed the experiments where 7 GENEVA flows or small/large Static-FEC flows or DP-FEC flows with different data transmission rates compete with long-lived TCP flows. We set the number of GENEVA flows transmitting data packets at a rate of 30 Mbps to 5, and the other two flows have a rate of 60 Mbps and 90 Mbps respectively. The reason why we set 5 flows with a data rate of 30 Mbps is that GENEVA with a lower data rate tends to aggressively increase the $Fwnd$ and fluctuate network conditions. As shown in **Table 2**, the average number of bursty packet loss events of the GENEVA flows was suppressed to about 83.7, and the average data loss rate decreases to 0.1%. Whereas the $TPindex$ of GENEVA flows becomes almost the same as that of small/large Static-FEC flows, both the total average number of bursty packet loss events and the total residual data loss rate of the GENEVA flows become lower than that of the small/large Static-FEC flows. In the case of DP-FEC flows, the total average residual data loss rate was not fully suppressed and become 0.46% by their aggressively increased $Fwnd$ (the total average $Fwnd$ of 44.0). In addition, the $TPindex$ of the DP-FEC flows



(a) Increased FEC window size



(b) Packet loss rate on the bottleneck link

Fig. 12 Trace of FEC window size of three selected GENEVA flows with different data transmission rates under RTT 10 ms, and packet loss rate on the bottleneck link.

decreases by more than 0.1, compared to that of the GENEVA flows. **Figure 12** shows the trace of the increased *Fwnd* of the three selected GENEVA flows and the link loss rates. Although the GENEVA flow with a data rate of 30 Mbps increases the *Fwnd* more aggressively, the link loss rates remain stable.

5. Related Work

A number of mechanisms for adjusting the degree of FEC while cooperating with TFRC have been proposed. To improve the video and playback quality of MPEG video while maintaining fairness with competing TCP flows, Wu, et al. [21] developed an optimization mechanism that adjusts FEC redundancy and the data transmission rate under bandwidth constraints of the TFRC equation. Because the proposed mechanism depends largely on the TFRC throughput equation, it tends to suffer from a degradation of streaming quality without effectively utilizing network resources in higher BDP environments. On the other hand, GENEVA is not subject to TFRC rate to avoid a degradation of video quality while effectively utilizing available network resources by adjusting the FEC window size. Seferoglu, et al. [17], [18] focused on packet losses caused by TCP-induced congestion, and proposed the TFRC that decides the best allocation of the available TFRC rate between source and FEC packets, based on the congestion prediction derived by the correlation between packet losses and the estimated RTT fluctuation. However, such predictions of loss events by using delay information result in a severe quality degradation, especially in high bandwidth paths [15], [16].

On the other hand, using packet loss characteristic modeled with two-state Gilbert model, Bolot, et al. [35] proposed the mechanism that determines the degree of FEC together with TFRC. In the presence of packet losses due to channel errors rather than network congestion, FEC techniques and its effectiveness for end-to-end protocols were examined, and the adaptive FEC schemes were proposed to combat such packet losses [19], [20]. For video applications using TCP, Tsugawa, et al. [8] applied an adaptive FEC scheme to TCP in order to avoid throughput fluctuations which result in video quality degradation. The scheme utilizes the algorithm proposed in Ref. [11] and derives the congestion window size needed for achieving the required data transmission rate. It then determines the appropriate degree of FEC to maintain the required rate according to the packet loss conditions, which achieves higher performance than static FEC approach.

The adaptive rate control with Dynamic FEC mechanism was proposed [33]. This mechanism tries to estimate network conditions while recovering lost data packets by fully added FEC redundancy. Since it focuses on the quality improvement for its own flow by aggressively increasing FEC, it holds the potential for adversely increasing traffic congestion and disturbing other communication qualities. To cooperate with other competing flows, the dynamic probing FEC (DP-FEC) [29] attempts to minimize the impact of FEC on other competing flows while utilizing the network resource effectively to optimize the recovery of lost data. Using variations in the intervals between packet loss events as a network indicator, it estimates the degree of FEC impact on competing TCP performance while gradually increasing the degree of added FEC redundancy. When the estimated FEC impact exceeds the defined threshold, it immediately reduces the degree of FEC by half. Since DP-FEC increases the degree of FEC redundancy regardless of RTT lengths and consumption bandwidth, there is a possibility that competing DP-FEC flows increase further traffic congestion and become unstable. As shown in Section 4, the largely increased FEC redundancy degrades both the communication quality of competing TCP flows and recovery conditions for DP-FEC flows. Since GENEVA algorithm adopts the supporting window control, which adjusts the number of unacknowledged packets sent to the network, to converge at an appropriate equilibrium point under stable conditions, it considers observed RTT lengths and consumption bandwidth.

6. Conclusion and Future Work

In this paper, we proposed the GENEVA mechanism that achieves higher streaming quality for high-performance real-time streaming applications. To achieve high network utilization, the GENEVA mechanism allows the flow to maintain the moderate network congestion in which bursty packet loss event is suppressed to improve FEC recover capabilities through adjustment of the FEC window size. The rates of increase/decrease in the FEC window size using GMIAD algorithm is designed to minimize the adverse impact on competing TCP performance by effectively utilizing available bandwidth that TCP fails to copy, and to achieve stable conditions. We verified the efficiency of GENEVA using an NS-2 simulator, and recognized that GENEVA retains higher streaming quality while minimizing adverse effects on TCP performance.

In our future work, we will make deeper analysis of the GENEVA algorithm to optimally adjust the parameters (e.g., the scaling properties of the total window size and SYN time) for ever-changing network conditions. Then, we will implement an actual high-performance application equipped with GENEVA and evaluate the GENEVA mechanism competing with many

high-performance streaming and TCP flows in complex and high bandwidth networks. Such evaluation will further improve the GENEVA algorithm, and then accelerate the deployment of high-quality streaming applications over heterogeneous future networks.

References

[1] Akamai Technologies, Inc.: 3rd Quarter, 2011 The State of the Internet, available from ⟨http://www.akamai.com/stateoftheinternet/⟩.

[2] Cisco white paper, Forecast and Methodology, 2010-2015, available from ⟨http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html⟩.

[3] 2011-2012 Report of the ICFA-SCIC Monitoring Working Group, available from ⟨http://www-iepm.slac.stanford.edu/pinger/⟩.

[4] 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).

[5] 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).

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

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

[8] Tsugawa, T., Fujita, N., Hama, T., Shimonishi, H. and Murase, T.: TCP-AFEC: An adaptive FEC code control for end-to-end bandwidth guarantee, *Proc. International Packet Video Workshop* (*PV 2007*) (Nov. 2007).

[9] 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).

[10] Matsuzono, K., Detchart, J., Cunche, M., Roca, V. and Asaeda, H.: Performance Analysis of a High-Performance Real-Time Application with Several AL-FEC Schemes, *Proc. IEEE Local Computer Networks* (*LCN*) (Oct. 2010).

[11] Ahn, J.S., Hong, S.W. and Heidemann, J.: An adaptive FEC code control algorithm for mobile wireless sensor networks, *Journal of Communications and Networks*, Vol.7, pp.489–499 (Dec. 2005).

[12] 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).

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

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

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

[16] Prasad, R.S., Jain, M. and Dovrolis, C.: On the effectiveness of delay-based congestion avoidance for TCP, *IEEE/ACM Trans. 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. Multimedia*, Vol.12, No.8, pp.869–885 (2010).

[19] Taal, J.R., Langendoen, K., van der Schaaf, A., van Dijk, H. and Lagendijk, R.: Adaptive end-to-end optimization of mobile video streaming using QoS negotiation, *Proc. IEEE ISCAS*, Scottsdale, AZ (May 2002).

[20] Karande, S.S. and Radha, H.: Rate-constrained adaptive FEC for video over erasure channels with memory, *Proc. IEEE ICIP*, Hainan, China (May 2008).

[21] 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).

[22] 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).

[23] Marcondes, C., Persson, A., Sanadidi, M.Y., Gerla, M., Shimonishi, H., Hama, T. and Murase, T.: Inline Path Characteristic Estimation to Improve TCP Performance in High Bandwidth-delay Networks, *Proc. PFLDnet'06* (Feb. 2006).

[24] Zhan, M., Dongsheng, Z. and Jihong, L.: A survey of Optical Networks and its visual management and cotrol, *Proc. Advanced Computer Control* (*ICACC*), Harbin, Chnia (2011).

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

[26] Gu, Y., Hong, X. and Grossman, R.L.: Experiences in Design and Implementation of a High Performance Transport Protocol, *Proc. ACM/IEEE Supercomputing 2004* (*SC2004*), Pittsburgh, PA, USA (Nov. 2004).

[27] Ngamwongwattana, B. and Sombun, A.: On Cross-Induced Congestion in Media Streaming, *Proc. Communications and Information Technologies* (*ISCIT*), Vientiane, Lao PDR (Oct. 2008).

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

[29] Matsuzono, K., Asaeda, H. and Murai, J.: DP-FEC: Dynamic Probing FEC for High-Performance Real-Time Interactive Video Streaming, *Journal of Information Processing*, IPSJ, Vol.20, No.1, pp.185–195 (2012).

[30] Jiang, H. and Dovrolis, C.: Why is the Internet Traffic Bursty in Short Time Scales ?, *Proc. ACM SIGMETRICS*, Vol.33, No.1, pp.241–252 (2005).

[31] Prasad, R.S., Dovrolis, C. and Thottan, M.: Router Buffer Sizing Revisited: The role of the output/input capacity ratio, *Proc. ACM CoNEXT*, USA (Dec. 2007).

[32] Mathis, M., Semke, J., Mahdavi, J. and Ott, T.: The macroscopic behavior of the TCP congestion avoidance algorithm, *Proc. ACM Compt. Commun.*, Vol.27, No.3, pp.67–82 (July 1997).

[33] 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).

[34] 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).

[35] 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).

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

[37] Floyd, S.: Highspeed TCP for Large Congestion Window, *IETF RFC* 3649 (Dec. 2003).

[38] Padhye, J., Firoiu, V., Towsley, D. and Kurose, J.: Modeling TCP Throughput: A Simole Model and its Empirical Validation, *Proc. ACM SIGCOMM* (Sep. 1998).

[39] Kleinrock, L.: *Queuing Theory*, Wiley, New York (1975).

[40] 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. Networking*, Vol.16, No.3, pp.628–641 (June 2008).

Kazuhisa Matsuzono received his B.E. from Keio University in 2005. He received 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.

Osamu Nakamura received a B.S. from Keio University in 1982, an M.S. in 1984, and a Ph.D. in engineering in 1993. He became Assistant Professor in the Faculty of Environment and Information Studies at Keio University's Shonan Fujisawa Campus in 1993, Associate Professor in 2000, and Professor in 2006. He has been Associate Director of the Auto-ID Labs Japan since 2003.

Jun Murai is Professor Faculty of Environment and Information Studies, Keio University. He received his M.E. and Ph.D. in computer science from Keio Univeristy 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.