Regular Paper

Implementing TCP-Friendliness in Digital Video over IP

Akimichi Ogawa,† Kazunori Sugiura,†† Osamu Nakamura† and Jun Murai†

In this paper, we implemented TCP-friendliness on DV (Digital Video)/RTP (Real-time Transportation Protocol). DV is a constant bit rate video stream that does not use inter-frame compression scheme. Without this implementation, DV/RTP creates constant bit rate stream, and consumes constant amount of network bandwidth. During times of network congestion, DV/RTP can consume unfairly large ratio of bandwidth compared to TCP. The TCP-friendliness in DV/RTP is obtained by dynamically discarding picture frames. By reducing use of network bandwidth during network congestion, our TCP-friendly mechanism reduces packet loss and the use of its network bandwidth is fair.

1. Introduction

Most Internet aware application uses TCP[1] as a transport layer protocol. TCP has a congestion control mechanism that controls amount of outgoing traffic. Congestion control mechanism enables collaboration with other flows, and reduces packet loss for its own flow. The congestion control mechanism for TCP is a very important factor of the Internet. Without this mechanism, the Internet can lead to a congestion collapse.

On the other hand, UDP[2] does not have a congestion control mechanism. UDP will send packets with the same rate even when the network is congested. TCP reduces sending rate when congestion occurs, TCP will decrease the window size during the network congestion. The decrease of window size will result to decrease of bandwidth usage. When heavy congestion is caused by the UDP traffic, TCP will suffer a massive packet loss and will receive very low bandwidth. Moreover, massive packet loss can lead to death of a TCP session.

Most Internet real-time applications use UDP as the transport layer protocol. The TCP resends data packets when it is lost. However, most real-time applications does not need to resend data packets. Rather than resending a packet, real-time applications require promptness of the data transmission. Moreover, TCP can not send multicast packets. For sending data to multiple receivers, multicast is required.

For fairness of bandwidth usage, congestion aware, TCP-friendly UDP traffic was proposed by S. Floyd[3]. However, real-time application that is TCP-friendly is not deployed and not widely used.

In this research, we designed and implemented TCP-friendly real-time video and audio stream, with a congestion control mechanism. We used DV (Digital Video)[4] as a real-time video and audio media. DV is a format for recording television informations within magnetic tape as digital information. In DV format, data transmission over IEEE1394[5] bus, which is a high speed serial bus, is also defined.

We used DVTS (Digital Video Transport System)[6]~[8] as an application that sends DV data using the Internet. DVTS consists of a IEEE1394 device driver[9] for FreeBSD[10], sender application, and receiver application. Since DVTS uses only consumer products, it does not require especial equipments. For interoperability with other DV over IP applications, DVTS uses RTP[11],[12].

The bandwidth share of the implemented system is evaluated in this paper. We evaluated share of bandwidth between DVTS with DVTS. We also evaluated DVTS traffic with TCP traffic.

2. Related Research

Related reserach about TCP friendly UDP mechanism are shown in this section.

The unfairness of flows like UDP, that does not have a congestion control mechanism, and the ways to detect them was proposed by S. Floyd[3]. In that paper, it is proposed that flows without congestion control mechanism are unfair, compared with TCP flows. UDP does not care about congestion, and sends data pack-

† Graduate School of Media and Governance, Keio University
†† CRL (Communications Research Laboratory)

ets continuously during congestion. However, during the congestion, TCP decreases its window size, and decreases the sending rate of data packets. Thus, in a congested network, UDP has advantage compared with TCP. In the Internet, which is a shared network, this advantage is not fair. S. Floyd also proposed that massive increasement of flow without congestion control can breakup the Internet by congestion. Moreover, S. Floyd proposed adding a mechanism for identifing flows without congestion control, and execute restriction to those flows. Identification of flow without congestion control is done by estimating TCP's maximum throughput. S. Floyd used Eq. 1 for estimation of TCP's maximum throughput. Flows that use much bandwidth than Eq. 1 is consuming much bandwidth than TCP flows. Flows that is consuming much bandwidth than estimated, is defined as "TCP-unfriendly flow".

$$T < \frac{1.5\sqrt{2/3} \times B}{R \times \sqrt{p}} \quad (1)$$

T : traffic quantity
B : MTU of connected link
R : RTT (Round Trip Time)
p : current packet loss ratio

There are various research about TCP-friendly UDP flows.

J. Mahdavi proposed a way to implement TCP-friendly flows[13] using the TCP throughput Eq. 1. Using Eq. 1, the maximum throughput of TCP flows can be estimated[3]. There, it is mentioned that UDP must not consume network bandwidth than TCP. When congestion does not occur, UDP flows may send as much packets as prefered. During the congestion, when the amount of packet loss is small enough, UDP applications can still send as much packets as perfered. However, when the amount of packet loss becomes large, UDP applications must reduce the consumption of network bandwidth into half. During times packet loss is not detected, the UDP application may increase the window size to one packet size for each RTT time. However, an ACK is sent for every UDP packet. Many real-time video transmission applications send large number of packets, and sending ACK for each of them can waste much bandwidth.

D. Sisalem proposed a LDA (Loss Delay Based Adjustment algorithm)[14]. In LDA, the sender controls the data sending rate, using the packet loss ratio and the RTT between the receivers. LDA uses RTP for sending data packets, and RTCP for feedback. Information for resolving packet loss ratio and RTT are included in RTCP feedback. The sender in LDA, decreases the window size when a packet loss is detected. When packet loss is not detected, the sender in LDA calculates a window called AIR (Additive Increase Rate). The sender in LDA increases window size calculated as AIR. Using the LDA algorithm, implementation of TCP-friendly UDP application is possible. However, when using CBR (Constant Bit Rate) application, simply decreasing the window size will lead to send buffer overflow in the sender host. Moreover, for real-time video transmission, it is very difficult to adjust the sending data size to the window size.

3. Design

In this research, we implemented TCP-friendliness within DVTS traffic, which is a CBR (Constant Bit Rate) traffic. TCP-friendliness for DVTS traffic will be obtained by estimating the throughput TCP traffic will get in the same situation. When DVTS traffic is getting unfair throughput, DVTS traffic will be reduced by discarding picture frames. The estimation of TCP throughput will be done by using Eq. 1.

When there are no packet loss in TCP traffic, TCP will increase the window size for the traffic. In this research, we assume that no packet loss is a sign of available network bandwidth.

Picture frame rate in DVTS is not proportional to bandwidth consumed by DVTS. The picture frame rate and the bandwidth consumed is shown in **Fig. 1**. The horizontal axis in Fig. 1 shows the picture frame rate. The vertical axis shows the consumed bandwidth in Mbps. The consumed bandwidth in DVTS gets larger rapidly when the picture frame rate is larger. When the picture frame rate is small, the effect by the change of picture frame rate is small. However, when the picture frame rate is large, the effect by the change of picture frame rate is large. For example, when changing the picture frame rate from 1/5 to 1/4, the consumed bandwidth increases 1.5 Mbps. When changing the picture frame rate from 1/2 to 1/1, the consumed bandwidth increases 15 Mbps.

Thus, simply increasing the picture frame

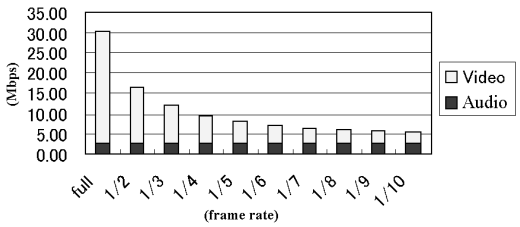

Fig. 1 Picture frame rate and consumed bandwidth in DVTS.

rate by the sign of no packet loss, can lead to rapid burst traffic. A scheme for changing picture frame rate in a inverseproportional manner is required.

In this research, we added a scheme for enlarging picture frame rates. In this scheme, it would be difficult for larger picture frame rates to be larger. We assume that a RTCP report with no packet loss reports available bandwidth. When a DVTS sender application receives a continuous RTCP report with no packet loss, the DVTS sender increases picture frame rate.

In this research, we use Eq. 2 for increasing picture frame rates. Using this equation, it would be difficult for larger picture frame rates to enlarge its picture frame rate. The difficulty for increasing picture frame rate can be adjusted by the parameter a in Eq. 2. The most appropriate value for parameter a is our future issue.

$$\frac{(f \times n)}{a} > 1 \qquad (2)$$

f : current picture frame rate
n : continuously received RTCP packets that reports 0 packet loss
a : parameter for increasing picture frame rate

3.1 Information Feedback Using RTCP

In this paper, we use RTCP for information feedback from the receiver application. The sender application receives information about state at the receiver application, from RTCP packets.

- number of packet loss
- RTT (Round Trip Time)

In RTP, the sender periodically sends RTCP SR (Sender Report) message, which describes the status within the sender. The receiver periodically sends RTCP RR (Receiver Report) message, which describes the status within the receiver.

The number of lost packets can be obtain by "cumulative number of packets lost" and "fraction lost" field of RTCP RR message. The "cumulative number of packet lost" field describes number of packet loss using 3 bytes. The 1 byte long "fraction lost" field shows packet loss with granularity of 1/256. To use information in the "fraction lost" field for the TCP throughput estimation, the information must be changed into 1/100 granularity. Since "fraction lost" field consists of 1 byte, RTCP RR can only report packet loss in granularity of 0.4%.

RTT between the DVTS sender application and receiver application can be obtain by using "LSR (Last Sender Report timestamp)" and "DLSR (Delay since Last Sender Report)" field of RTCP RR. LSR field in RTCP SR packet shows the time the packet was sent from the sender. The time is a time of the sender. The receiver calculates the time difference between the received time of RTCP SR and the send time of RTCP RR. The calculated time is shown as the "DLSR" field of RTCP RR. RTT can be calculated at the sender by using Eq. 3. The calculation of RTT will be done when a sender receives a RTCP RR packet. The RTT is calculated by subtracting LSR and DLSR from current time at the sender. In this research, we use these informations from RTCP.

$$RTT = t - t_{DLSR} - t_{LSR} \qquad (3)$$

RTT : Round Trip Time
t : current time
t_{DLSR} : DLSR time
t_{LSR} : LSR time

4. Evaluation

The following 3 requirement is needed in network adaptive TCP-friendly system.

1) Packet loss must be decreased by using the TCP-friendly system.
2) Network bandwidth must be distributed fairly. When multiple flows exists within the network, each flows must use same amount of network bandwidth. Massive use of network bandwidth by particular flow is unfair.
3) Network bandwidth must be used efficiently. When other flow does not exist,

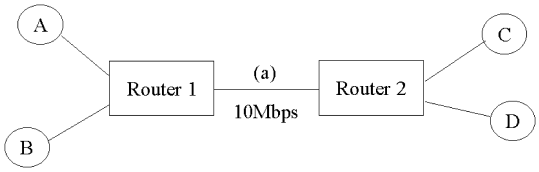

Fig. 2 Network topology for evaluation.

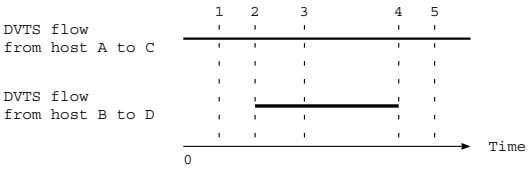

Fig. 4 Timing of DVTS flows.

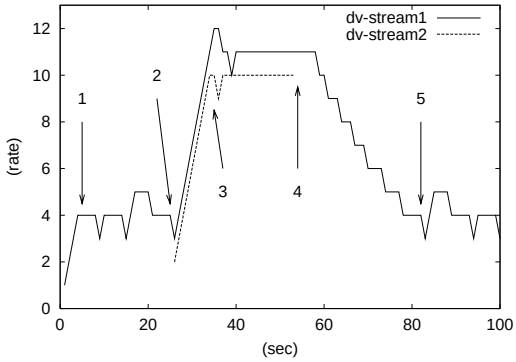

Fig. 3 Change of picture frame rate with multiple DVTS flows.

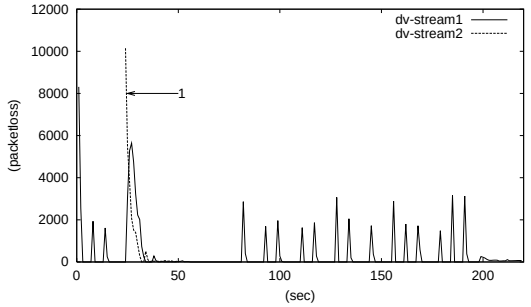

Fig. 5 Packet loss at each DVTS flow.

the existing flow must have the ability to use as much bandwidth as possible.

For evaluation, we have done two tests.

1) DVTS flow with DVTS flow
2) DVTS flow with TCP flow

4.1 Collaboration between Multiple DVTS Flows

Evaluation for multiple DVTS flows were done with topology shown in **Fig. 2**. Host AB and CD shares the same 10 Mbps link (a). In this evaluation, we sent adaptive DVTS flows from host A to C and host B to D. First, DVTS flow from host A to C was sent. After DVTS flow from host A to C is converged, we added a new flow from host B to D. Next, after both DVTS flows are converged, we stopped DVTS flow from host B to D. DVTS version 0.3.6 was used for this evaluation.

Figure 3 shows picture frame rate for each DVTS flow. The horizontal axis of Fig. 3 shows sequence number of RTCP reports. In this evaluation, RTCP reports are sent each 3 seconds. The vertical axis of Fig. 3 shows the picture frame rate of each DVTS flow. The timing of the DVTS flows are shown in **Fig. 4**. The picture frame rate of first DVTS flow is converged around 1/4 at point 1 of Fig. 3. Picture frame rate 1/4 is the largest picture frame rate under 10 Mbps. Thus, DVTS uses as much bandwidth as possible when there are no other traffic.

Next, a new DVTS flow is added at point 2 of Fig. 3. Each DVTS flow reduced their picture frame rate, after a new DVTS traffic was added to the network. At point 3 of Fig. 3, the picture frame rate of the previous flow converged at 1/11. The picture frame rate for the new flow converged at 1/10. Picture frame rate 1/10 consumes about 4.8 Mbps. Since the network bandwidth of link (a) is 10 Mbps, this evaluation shows that multiple DVTS flows shares network bandwidth, and consumes as much bandwidth as possible.

At point 4 of Fig. 3, the DVTS added at point 2 is stopped. After point 4, the previous DVTS flow increases its picture frame rate.

Finally, at point 5 of Fig. 3, the picture frame rate for the previous DVTS flow is back to 1/4. This evaluation shows, that DVTS recovers use of network bandwidth when there are no other traffic.

Figure 5 shows the packet loss of each DVTS flow. The horizontal axis of Fig. 5 shows sequence number of RTCP reports. The vertical axis of Fig. 5 shows the number of packets loss between the RTCP reports. At point 1 of Fig. 5, there are remarkable packet loss at DVTS flow from host A. Because of the packet loss, DVTS from host A reduces picture frame rate. When a new DVTS flow from host B is added, there are packet loss at both DVTS flows. Both DVTS flows reduce picture frame rate after the packet loss. After both DVTS flows are converged, number of packet loss is decreased for both DVTS flows.

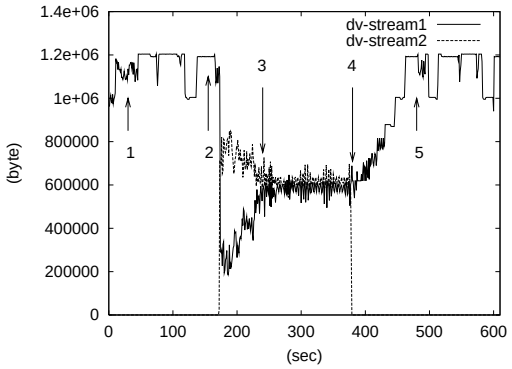

Fig. 6 Traffic pattern of each DVTS flow.

Figure 6 shows the network bandwidth consumed by each DVTS flow. The horizontal axis shows time in seconds. The vertical axis shows the consumed network bandwidth in bytes. At point 1 of Fig. 6, there are only 1 DVTS flow. When there are only 1 DVTS flow, it consumes as much bandwidth as possible. Next, at point 2 of Fig. 6, a new DVTS flow is added. When a new DVTS flow is added, both DVTS flow reduces use of network. At point 3 of Fig. 6, both DVTS flow consumes same amount of network bandwidth. DVTS flow from host B is stopped at point 4 of Fig. 6. After DVTS flow from host B is stopped, DVTS flow from host A increases use of network. Finally, at point 5, when only 1 DVTS flow is in link (a), DVTS flow from host A uses as much network bandwidth as point 1.

This evaluation shows that multiple DVTS flows shares network bandwidth, and consumes as much bandwidth as possible. When there are only one DVTS traffic, the DVTS traffic uses network bandwidth the most. When another DVTS traffic is sent over the same network, packet loss is detected at both DVTS traffics. Then both DVTS traffics decreases the sending picture frame rate and shares the available bandwidth. The sending picture frame rate is balanced by moderate packet loss.

4.2 Collaboration between DVTS Flow and TCP Flow

Evaluation of adaptive DVTS flow with TCP flow, was done with the network topology shown in **Fig. 7**. Hosts A, B, C and D, E, F shares link (a). Network bandwidth for link (a) is 10 Mbps. First, DVTS flow from host A is sent to host D. After picture frame rate for the DVTS flow is converged, FTP flow was added from host B to host E. The data sent by FTP was large enough, and FTP did not end during

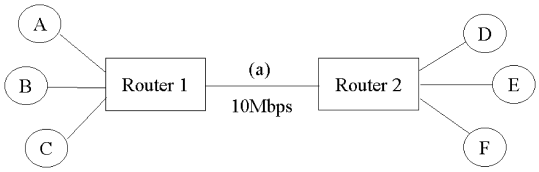

Fig. 7 Network topology for evaluation.

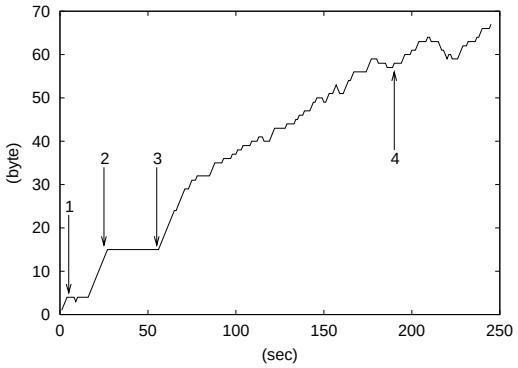

Fig. 8 Change of picture frame rate with DVTS flow and TCP flow.

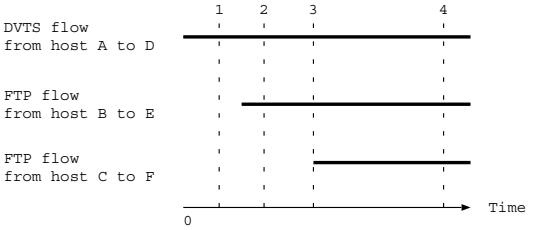

Fig. 9 Timing of DVTS and TCP flows.

the evaluation. Next, after the picture frame rate for the DVTS flow was converged again, we started another FTP flow from host C to host F.

Picture frame rate for the adaptive DVTS flow is shown in **Fig. 8**. The horizontal axis of Fig. 8 shows sequence number of RTCP reports. The vertical axis of Fig. 8 shows the picture frame rate. The timing of the DVTS and FTP flows are shown in **Fig. 9**. At point 1 of Fig. 8, there are only 1 DVTS flow. When there is only 1 DVTS flow, the picture frame rate for the DVTS flow was converged at 1/4. At point 2 of Fig. 8, a new FTP flow was added to the network. The picture frame rate for the DVTS flow converged at about 1/15 with 1 FTP flow. At point 3 of Fig. 8, another FTP flow was added to the network. With two FTP flows, picture frame rate for DVTS was converged around 1/60. This evaluation shows, that when a new TCP flow is added to the

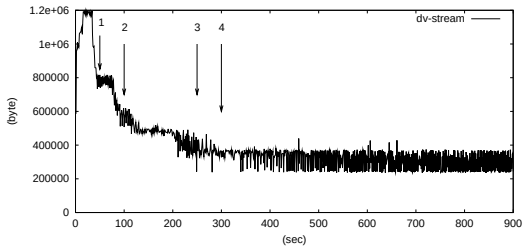

Fig. 10 Traffic pattern of DVTS flow and TCP flow (DVTS traffic).

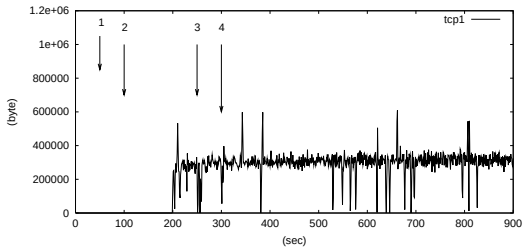

Fig. 11 Traffic pattern of DVTS flow and TCP flow (TCP traffic 1).

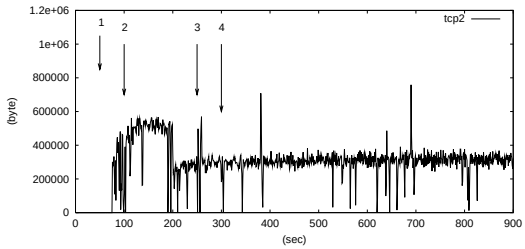

Fig. 12 Traffic pattern of DVTS flow and TCP flow (TCP traffic 2).

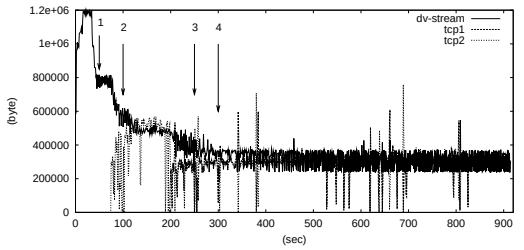

Fig. 13 Traffic pattern of DVTS flow and TCP flow.

network, adaptive DVTS decreases its picture frame rate.

The network bandwidth used by each flow is shown in **Fig. 10**, **11**, **12** and **13**. The horizontal axis shows time in seconds. The vertical axis shows the consumed network bandwidth in bytes. At point 1 of Fig. 13, only 1 DVTS flow is in the network. At that point, DVTS uses as much bandwidth as possible. After a new FTP flow was added, DVTS flow shared bandwidth with the new FTP flow. FTP and DVTS consumed same amount of bandwidth at point 2 of Fig. 13. The bandwidth DVTS used at this point was half of point 1. When FTP flow becomes 2, at point 3, DVTS decreased its network use. At point 4, each flow uses same amount of bandwidth. The bandwidth DVTS used at this point was 1/3 of point 1. This evaluation shows, that adaptive DVTS shares network bandwidth fairly with TCP.

This evaluation shows that DVTS flow and TCP flow shares network bandwidth. Moreover, both TCP and DVTS flows consumes as much bandwidth as possible. When there are only one DVTS traffic, the DVTS traffic uses network bandwidth the most. When a TCP traffic is sent over the same network, packet loss is detected at both TCP and DVTS traffic. Then both TCP and DVTS traffics decreases the sending picture frame rate and shares the available bandwidth. The sending picture frame rate is balanced by moderate packet loss.

The evaluations in this section shows that DVTS, we developed, adapts to current network. Thus, the adaptive DVTS implementation is mathmatically TCP-friendly, as proposed in the technical note[13].

However, this scheme requires packet loss to be detected to trigger the adjustment mechanism. The packet loss on real-time video and audio transmission will result to noise on the video and audio. Thus, TCP-friendly real-time video and audio transmission applications are not deployed. Mathmatically making the real-time video and audio transmission TCP-friendly can be realized. However, since packet loss can decrease the quality of video and audio drastically, a bandwidth estimation mechanism that can work before large number of packets are lost, is required.

5. Conclusion

Most real-time application uses UDP as a transport layer protocol. UDP does not have a congestion control mechanism. Thus, UDP's coexistence with TCP, which has a congestion control mechanism, is being impossible. To resolve this issue, application level congestion control is required, when using UDP as a transport layer protocol.

In this research, we focused to DV as a real-time consumer video and audio media, and created multimedia video communication tool (DVTS) with TCP-friendly congestion control.

By implementing congestion control mechanism within DVTS, TCP-friendly real-time video and audio transmission is realized. DVTS will use as much bandwidth as possible when there are no other traffic within the network. However, when congestion occurs within the network, it reduces use of network bandwidth. Reduction of network bandwidth is done by discarding picture frames from the sender application dynamically. By reducing use of network bandwidth during network congestion, DVTS is TCP-friendly, and the use of network bandwidth will be fair.

However, this scheme requires packet loss to be detected to trigger the adjustment mechanism. The packet loss on real-time video and audio transmission will result to malformed output on the video and audio. Thus, TCP-friendly real-time video and audio transmission applications are not deployed. Mathmatically making the real-time video and audio transmission TCP-friendly can be realized. However, since packet loss can decrease the quality of video and audio drastically, a bandwidth estimation mechanism that can work before large number of packets are lost, is required.

References

1) Postel, J.: *Transmission Control Protocol*, RFC 793 (Sep. 1981).
2) Postel, J.: *User Datagram Protocol*, RFC 768 (Aug. 1980).
3) Floyd, S. and Fall, K.: Promoting the use of end-to-end congestion control in the Internet, *IEEE/ACM Trans. Networking* (Aug. 1998).
4) Specifications of Consumer-Use Digital VCRś using 6.3 mm magnetic tape, *HD Digital VCR Conference* (1994).
5) IEEE Standard for a High Performance Serial Bus, *IEEE Computer Society* (1995).
6) Ogawa, A.: *DVTS (Digital Video Transport System) WWW page*, WWW page (Nov. 2001).
7) Ogawa, A., Kobayashi, K., Sugiura, K., Nakamura, O. and Murai, J.: Design and Implementation of DV stream over Internet, *IWS99* (Feb. 1999).
8) Ogawa, A., Kobayashi, K., Sugiura, K., Nakamura, O. and Murai, J.: Design and Implementation of DV based Video over RTP, *Packet Video 2000* (May 2000).
9) Kobayashi, K.: Design and Implementation of Firewire Device Driver on Freebsd, *USENIX 1999*, pp.41–51 (1999).
10) *FreeBSD Home-page* (2000). http://www.freebsd.org/
11) Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V. (Audio-Video Transport Working Group): *RTP: A Transport Protocol for Real-Time Applications*, RFC 1889 (Jan. 1996).
12) Kobayashi, K., Ogawa, A., Casner, S. and Bormann, C.: *RTP Payload Format for DV Format Video*, Internet Draft (June 2000).
13) Mahdavi, J. and Floyd, S.: *TCP-friendly unicast rate-based flow control* (Jan. 1997). Technical Note available from http://www.psc.edu/networking/papers/tcp_friendly.html
14) Sialem, D. and Schulzrinne, H.: The loss-delay based adjustment algorithm: A tcp-friendly adaption scheme, *Network and Operating System Support for Digital Audio and Video* (*NOSSDAV*) (July 1998).

(Received June 1, 2001)
(Accepted November 14, 2001)

Akimichi Ogawa received the B.S. degree in Enviromental Information from Keio University in 1998. In 2000, he received the M.S. degree in Media and Governance from Keio University. He has been researching the Internet technologies. His research interests include high-speed networks and real-time multimedia communication.

Kazunori Sugiura Resercher of Communication Research laboratory, Information and Network System Division, High Speed Network Group. Received B.S. degree in Environmental Information from Keio Univ. in 1994, M.S. degree in Media and Governance 1996, Entering Doctorial Degree in 1996. Speciality in Broadband Network in real-time multimedia and ubiquitous environment.

Osamu Nakamura received the B.S. degree and M.S. degrees in mathematics from Keio University in 1983 and 1985. From 1990 to 1993, he was an assistant professor at The University of Tokyo. From 1993 to 2000 he was an assistant professor at Keio University, Faculty of Environmental Information. From 2000 he has been an associate professor at same department. He received Ph.D. degree from Keio University in 1990. He has been a Board Member of WIDE Project since 1988. He has been researching the Internet technologies. His research interests include operating system, high-speed networks, and network management.

Jun Murai Professor, Faculty of Environmental Information, Keio University. Born in March 1955 in Tokyo. Graduated Keio University in 1979, Department of Mathematics, Faculty of Science and Technology, M.S. for Computer Science from Keio University in 1981, received Ph.D. in Computer Science, Keio University, 1987. Director, Keio Research Institute at SFC. The President of Japan Network Information Centre (JPNIC). The Internet Corporation for Assigned Names and Numbers (ICANN) Board of Director. Adjunct Professor at Institute of Advanced Studies, United Nations University. He also teaches at Tokyo University of Art and Music. Specialized in computer science, computer network, and computer communication. His recent publications include "Internet II" (Iwanami Publication), July 1998, "Evolution and Revolution of the Internet in Japan" (Proc. of CyberJapan: Technology, Policy Society Symposium, The Library of Congress, May, 1996).