Recommended Paper

Video Transfer Protocol over Internet with Congestion Control Based on Two Level Rate Control

Teruyuki Hasegawa,† Toru Hasegawa† and Toshihiko Kato†

Video data transfer is one of the major type of traffic over the Internet. However, it is impossible for the current Internet to guarantee the network QoS for video transfer, and an influx of a large amount of video data into the Internet may cause serious network congestion. To resolve these problems, we previously proposed a protocol using congestion control based on rate control of video coding level and data transfer level, and evaluated this protocol using software simulation. However, in order to apply this protocol to real video communication, our previous protocol had some problems mainly with regard to efficiency. In this paper, we present an improved specification of a video transfer protocol with two level rate control. This paper also describes the results of implementation and evaluation of an actual video transfer system using our improved protocol, and a commercial video tool with H.261 encoding. The results show our protocol can transfer video data effectively and fairly through a congested network.

1. Introduction

Recently, video data transfer has become a major type of traffic over the Internet. In video traffic, individual video data packet needs to be delivered before the time when the packet is scheduled to be reproduced, and in order to satisfy this requirement, it is expected that a network will guarantee the Quality of Service of network (*network QoS* or *QoS* for short) such as bandwidth and delay variation for video traffic. However, the current Internet cannot provide the QoS guarantee, so all traffic tries to use all of the available bandwidth by itself. Therefore, if the sum of generated traffic becomes larger than the network bandwidth, delay and/or packet loss will occur at the sender terminals and the intermediate systems (routers). This situation is network congestion.

Therefore, in order to transfer video data over the Internet, a congestion control with the following functions is required.

- The sender and receiver terminals support more than one coding rate for video data.
- If QoS is degraded due to network congestion, the sender terminals detect this QoS degradation and decrease the coding rate of video data so that the video traffic is not affected by the congestion.
- If network congestion has ended and QoS has improved, the sender terminals detect this QoS improvement and increase the coding rate to the maximum rate at which data can be transferred under the improved QoS.

There are several proposals on congestion control for video data transfer [1)~3)], where the sender terminal changes the coding rate based on packet loss ratio and/or RTT (Round Trip Time). Further, Refs. 4) and 5) propose the TCP friendly congestion control, which aims to make all traffic in the Internet including video traffic use a TCP-like congestion control mechanism. These congestion control schemes decrease and increase the coding rate of video data itself, in order to detect the start and end of network congestion. However, in the case where multiple video data traffic is multiplexed and causes network congestion, these schemes may cause an oscillation where the coding rate increases and decreases continuously. The reason for this oscillation is as follows. If network congestion occurs, all or most of the multiplexed traffic will start decreasing the rate. After that, network congestion ends, the multiplexed traffic starts to send more data by increasing the coding rate, and this causes network congestion again. As a result, the coding rate keeps decreasing and increasing instead of using the average of the available bandwidth. This situation is not good for the quality of

† KDDI R&D Laboratories Inc.

The initial version of this paper was presented at the DPS workshop held on Dec. 1999, which was sponsored by SIGDPS. This paper was recommended to be submitted to the Journal of IPSJ by the chairperson of SIGDPS.

video data itself[6].

In order to resolve this problem, we previously proposed a video transfer protocol using a congestion control method with a two level rate control[6].

- By rate control of data transfer level, degradation and improvement of QoS are detected. At this level, a TCP-like congestion control is adopted in order to detect the QoS change quickly (e.g., in the order of RTT), and to allow the video traffic to behave similarly to TCP traffic.
- By rate control of video coding level, an appropriate coding rate is selected by observing the fluctuation of the data transfer. This coding rate is adjusted with a duration long enough (e.g., a few seconds) not to degrade the quality of video itself.

In Ref. 6), we described the detailed procedure of our protocol and the results of an evaluation using software simulation. However, in order to apply this protocol to real video communication, our previous protocol had the following problems.

(1) Since our previous protocol uses one acknowledgment packet (ACK) for each data packet (DT) for RTT measurement, too many ACKs are generated.

(2) Since our previous protocol uses RTT, the estimation of QoS is affected not only by the transmission delay in the DT forwarding direction, but also by that in the reverse direction.

(3) In order to check whether there is no network congestion, the sender needs to transmit DTs at a transmission rate determined according to the protocol, even if there is no video data to be transmitted. In such a case, DTs carry unnecessary padding data.

(4) Since the coding rate and the DT transmission rate are determined independently, it is possible that the transmission rate of DTs becomes much lower than the coding rate. In such a case, a large buffering delay is inserted because a large amount of video data waiting for transmission is accumulated.

Moreover, Ref. 6) evaluated our previous protocol only through a software simulation without using any video data. In order to verify that the protocol can be applied to real video communication, it is required to implement a video transfer system with such a protocol and to evaluate the performance of the system using actual video data.

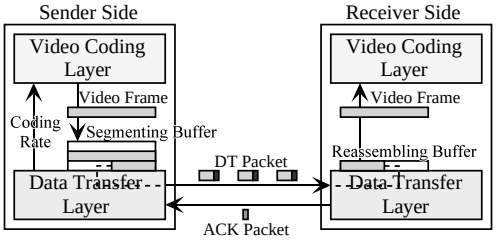


Fig. 1 Layer structure.

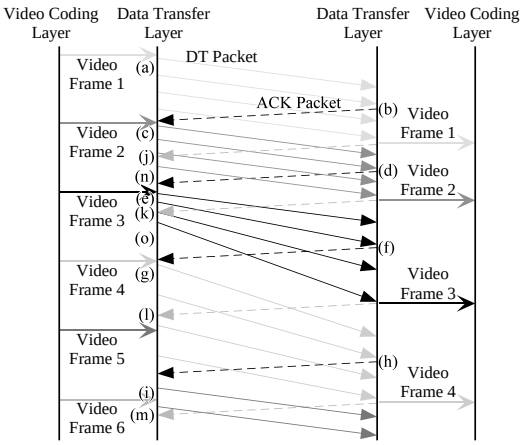


Fig. 2 Communication sequence.

In this paper, we present a video transfer protocol which resolves the problems of our previous version, and describe the results of implementation and evaluation of an actual video transfer system using our improved protocol and a commercial video tool with H.261[7] encoding. This paper is organized as follows. Sections 2 and 3 explain the specifications of our improved protocol. Section 4 describes the implementation of the video transfer system with our protocol. Section 5 evaluates the performance of the protocol using the video transfer system. Section 6 gives some discussions on the results. Section 7 concludes this paper.

2. Overviews

Figures 1 and **2** show the layer structure of our improved protocol and an example of a communication sequence, respectively. The overview of the protocol is as follows[8),9)].

(1) As shown in Fig. 1, on both the sender and receiver sides, the protocol is composed of two layers; one is a *video coding layer* and the other is a *data transfer layer*. On the sender side, the video coding layer encodes video data

on a per frame basis at the rate indicated by the data transfer layer, and stores a video frame with its encode time in the segmenting buffer. The data transfer layer segments a video frame into DT packets and transmits them. On the receiver side, the data transfer layer reassembles DT packets into a video frame, and delivers it to the video coding layer.

(2) The data transfer layer uses DT packets and ACK packets. A DT includes a segmented frame whose length is fixed, and some control fields such as timestamp, group number, sequence number, and ACK request flag. The timestamp indicates the transmission time of the DT. The group number and sequence number are used to identify a DT. The group number is incremented after requesting an ACK, or after changing the transmission rate of DTs, or on sending the first DT for a video frame. The sequence number is incremented on each DT transmission and reset to 0 when the group number is updated. The ACK request flag indicates that the receiver side needs to transmit an ACK in response to this DT.

On the other hand, an ACK includes group number, sequence number, average one-way delay and packet loss ratio. The group number and the sequence number indicate those of the corresponding DT. The average one-way delay means the average of the difference between sending time and receiving time of DTs which have been received after the previous ACK. By using the one-way delay, the problem of QoS estimation by RTT is resolved. The packet loss ratio is the ratio of lost DTs among DTs which have been received after the previous ACK.

In order to avoid the frequent ACK problem in our previous protocol, the sender side requests an ACK every time after it has transmitted a specific number of DTs, or when it transmits a DT including the last segment in a frame (see Fig. 2 (a)/(b)).

(3) The sender side controls the DT transmission rate by changing the time interval of transmitting DTs (*DT transmission interval* for short) in the following way.

The sender side keeps the minimum value of the average one-way delay as the minimum one-way delay. If the difference between the average one-way delay in an ACK and the minimum one-way delay is smaller than a threshold, the sender side considers that there is no congestion, and decreases the DT transmission interval (see Fig. 2 (b)/(c), (d)/(e), (h)/(i)). On the other hand, in the case that the difference between the average one-way delay and the minimum one-way delay is larger than the threshold, the sender side considers that congestion occurs and increases the DT transmission interval (see Fig. 2 (f)/(g)). The increase and decrease of the DT transmission interval are controlled by separate algorithms for severe and moderate congestion, which correspond to the slow start and congestion avoidance algorithms in the TCP congestion control [10], respectively. Section 3.2 describes the details of these algorithms.

It should be noted that DTs are not transmitted in the DT transmission interval when there are no video frames stored in the segmenting buffer (see Fig. 2 (n)/(o)). This resolves the problem that unnecessary padding data is transmitted by our previous protocol.

It should be also noted that once the DT transmission interval is changed, it will not be changed again until the sender side receives an ACK for DTs having been transmitted by it (see Fig. 2 (j)–(m)). For this purpose, the sender side maintains the value of the group number when the DT transmission interval is changed and compares it with the group number in a received ACK.

(4) The sender side uses one coding rate value for a duration such as a few seconds. The data transfer layer calculates an appropriate coding rate for the next duration by observing the changes of DT transmission interval, and reports it to the video coding layer. The detailed algorithm on how to determine the coding rate is described in Section 3.3.

(5) Since the DT transmission intervals change dynamically, it is possible that the coding rate becomes larger than the DT transmission rate. In such cases, the data transfer layer on the sender side discards a whole frame from the segmenting buffer after a specific time (e.g., 100 msec order) has passed since the frame was encoded. This resolves the problem of our previous protocol that there is a large buffering delay inserted on the sender side.

3. Detailed Algorithms

3.1 Parameters

Our protocol uses the following parameters.

- $\boldsymbol{int}$: value of DT transmission interval.
- $\boldsymbol{int_{cong}}$: value of *int* when network congestion is detected.
- $\boldsymbol{int_{max}}$, $\boldsymbol{int_{min}}$: maximum and minimum

values of *int*, which are determined by the maximum and minimum coding rates, respectively.

- ***delay$_{min}$***: minimum value of average one-way delay reported by ACKs. On receiving an ACK, it is verified whether the value of the average one-way delay in the ACK is smaller than $delay_{min}$. If so, $delay_{min}$ is updated to the received value.
- $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\varepsilon}$: thresholds of average one-way delay and packet loss ratio, for detecting network congestion. When an average one-way delay minus $delay_{min}$ is larger than or equal to β, or a packet loss ratio is larger than ε, the congestion is considered to be severe, and when an average one-way delay minus $delay_{min}$ is between α and β, the congestion is considered to be moderate. When an average one-way delay minus $delay_{min}$ becomes smaller than α, it is considered that the congestion has ended or that there is no congestion.
- ***A***, ***B***: increasing ratios of *int* when moderate and severe network congestion is detected, respectively.
- ***a***, ***b***: coefficients for decreasing *int* while no congestion is detected. a is an increase of the number of DTs allowed to transmit in one unit time. It linearly increases the DT transmission rate, which is proportional to the reciprocal of *int*. b is the decreasing ratio of *int*. Either a or b is selected according to the values of *int* and int_{cong}.
- $\boldsymbol{\Delta}$, $\boldsymbol{\delta}$: duration of adjusting coding rate, and threshold ratio for determining the next coding rate.
- $\boldsymbol{\mu}$: coefficient to reflect the average of time intervals in which DTs have been actually transmitted, in determining the next coding rate.

3.2 Algorithm for Determining DT Transmission Interval

Figure 3 shows the pseudo code for determining the value of *int* on the sender side. According to the difference between the average one-way delay ($delay_{ave}$ in Fig. 3) and $delay_{min}$, and/or the value of packet loss ratio, the data transfer layer performs one of the following algorithms.

3.2.1 Algorithm for Severe Congestion

When the average one-way delay minus $delay_{min}$ is larger than or equal to β, or the value of packet loss ratio (*loss* in Fig. 3) is larger than ε, the value of *int* at this time is saved in

```
if((delay_ave - delay_min >= β)
    || (loss > ε)){              /* (a) */
    int_cong = int;
    int = int * B;
    int = MIN(int, int_max);
}
elseif(delay_ave - delay_min >= α){ /* (b) */
    int_cong = int;
    int = int * A;
    int = MIN(int, int_max);
}
elseif(delay_ave - delay_min < α){
    if(int > int_cong * A)       /* (c) */
        int = int * b;
    else                         /* (d) */
        int = int / (1 + int * a);
    int = MAX(int, int_min);
}
```

Fig. 3 Pseudo code for determining DT transmission interval.

int_{cong} and the value of *int* is multiplied by B (see Fig. 3 (a)). After that, if there is no congestion, that is, the average one-way delay minus $delay_{min}$ is smaller than α, *int* is decreased to $int \times b$ (see Fig. 3 (c)). Such a decrease of *int* causes an exponential increase of DT transmission rate, and we consider that it corresponds to the slow start algorithm used in TCP. If the decreased value of *int* is smaller than $int_{cong} \times A$, the data transfer layer follows the procedure described bellow (see Fig. 3 (d)).

3.2.2 Algorithm for Moderate Congestion

When the average one-way delay minus $delay_{min}$ is between α and β, the value of *int* at this time is saved in int_{cong} and the value of *int* is multiplied by A (see Fig. 3 (b)). After that, if there is no congestion, $1/int$ is increased to $1/int + a$ (see Fig. 3 (d)). This causes a linear increase of the DT transmission rate, and we consider that it corresponds to the congestion avoidance algorithm used in TCP.

3.3 Algorithm for Determining Coding Rate

The data transfer layer on the sender side determines the coding rate based on the changes of *int* during the duration Δ. It performs the following procedure at every end of Δ.

(1) The data transfer layer calculates the histogram of the number of DTs transmitted with various values of *int*. Let n, int_i, and s_i $(1 \leq i \leq n)$ be the number of values of *int* used in a duration Δ, the ith value of *int* in increasing order (i.e., $int_1 < \ldots < int_n$), and the num-

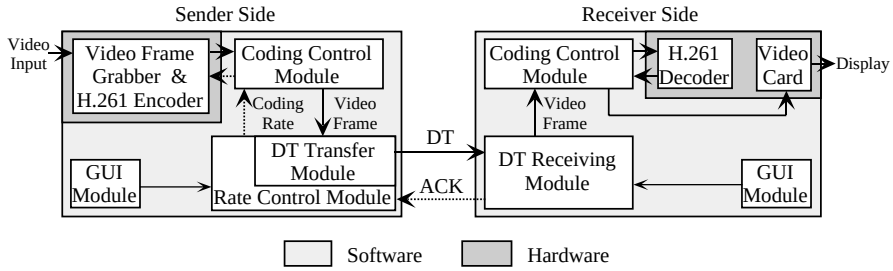


Fig. 4 Configuration of video transfer system.

ber of DTs which are transmitted with int_i, respectively. It should be noted that $\sum_{i=1}^{n} s_i$ is equal to the total number of DTs transmitted during the duration Δ.

(2) Then, the data transfer layer calculates the value of int such that δ of all transmitted DTs are transmitted at a smaller interval (at a higher rate) than that value. That is, it calculates the smallest value of k such that

$$\frac{\sum_{i=1}^{k} s_i}{\sum_{i=1}^{n} s_i} \geq \delta.$$

(3) The data transfer layer calculates int_{ave}, the average time interval of DTs which have been actually transmitted during this duration Δ, by the following equation.

$$int_{ave} = \frac{\Delta}{\sum_{i=1}^{n} s_i}$$

(4) Based on int_k, int_{ave} and μ, the time interval int_{next}, which is used for selecting the coding rate of the next duration Δ, is determined as follows. At first, int_{next} is set to int_k. Next, if int_{ave} is larger than $\mu \times int_{cur}$, which corresponds to the current coding rate of this duration Δ, and if int_{next} is smaller than int_{cur}, int_{next} is set to the same value of int_{cur}.

```
int_next = int_k;
if((int_ave > int_cur * μ) && (int_next < int_cur))
    int_next = int_cur;
```

(5) The data transfer layer calculates $rate_{next}$ as follows:

$$rate_{next} = \frac{plen \times 8}{int_{next}}$$

where $plen$ [byte] indicates the packet length in which the data transfer layer transmits DTs. Finally, among the predefined coding rates values, the data transfer layer selects the largest one being equal to or smaller than $rate_{next}$ and reports it to the video coding layer. The reported coding rate is used in the video coding layer during the next duration Δ.

4. Implementation

We have implemented a video transfer system including our proposed protocol on UNIX workstations. **Figure 4** shows its configuration. It runs on Sun workstations with a PCI video capture card (Sun Microsystems, SunVideo Plus) which provides a video frame grabbing function and a hardware H.261 codec function. In H.261, a video frame is divided into some blocks (*macroblock*), and either intra-frame or inter-frame compression is applied to each macroblock. In inter-frame mode, compression is applied to only the parts of a frame that have changed from previous frames in order to gain higher rate of compression. However, if any macroblock is corrupted or dropped, the corresponding macroblocks compressed in inter-frame mode cannot be decoded.

Figure 5 shows a display image of the sender system. The sender software is composed of the following modules.

- **Coding Control Module**
  This provides the functions of the video coding layer described in Section 2 together with SunVideo Plus. That is, it sets up H.261 coding parameters on SunVideo Plus such as *coding rate*, *frame rate*, and *maximum quantization* based on the coding rate indicated by the rate control module. The last parameter is used to specify the balance of the frame rate and the amount of bits per image (i.e., image quality). This module grabs an H.261 encoded video frame from SunVideo Plus and passes it to the DT transfer module. The control of SunVideo Plus is performed through the XIL interface [11),12)].

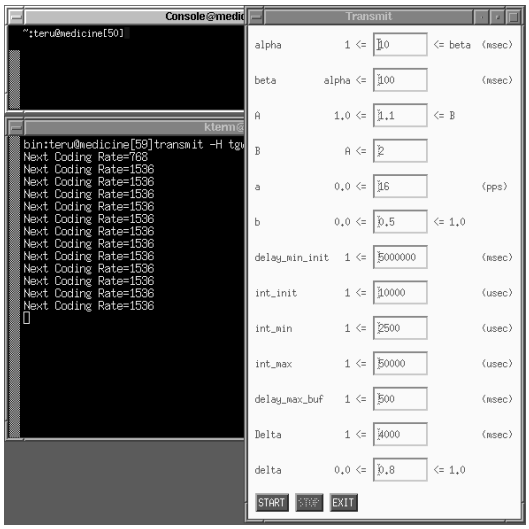


Fig. 5 Display image in sender system.

- **DT Transfer Module**
  This realizes the functions for segmenting a video frame into DTs and transmitting DTs according to the time interval directed by the rate control module.
- **Rate Control Module**
  This receives ACKs from the receiver side and realizes the functions for determining the value of the DT transmission interval and the coding rate for the next duration Δ as described in Section 3.
- **GUI (Graphical User Interface) Module**
  The user can start/stop the video transfer and terminate the whole program, through the *start*, *stop*, and *exit* buttons, respectively. In addition, the values of most protocol parameters described in Section 3.1 can be set from the GUI module.

In order to make the time interval between DT transmissions accurate, the coding control module, the DT transfer module, and the rate control module including the GUI module have been implemented separately as a POSIX compliant thread.

Figure 6 shows a display image of the receiver system. The receiver software is composed of the following modules.

- **Coding Control Module**
  This supports the functions of the video coding layer together with SunVideo Plus. When an H.261 encoded video frame is delivered from the DT receiving module, this module decodes it with the help of the H.261 decoder in SunVideo Plus and displays the decoded video frame through the video card. These decode and display operations are performed via the XIL interface.
- **DT Receiving Module**
  This realizes DT packet receiving/reassembling and measurement/reporting of the average one-way delay and the packet loss ratio.
- **GUI Module**
  This supports re-initialization and exit functions for the program through the *init* and *exit* buttons.

The receiver software has been implemented as one thread because H.261 video decoding, which is the only heavily loaded task on the receiver side, is performed on a hardware.

Fig. 6 Display image in receiver system.

5. Performance Evaluation

In order to evaluate the proposed protocol, we have performed some communication tests using the configuration shown in **Fig. 7**. In the evaluation, we used two pairs of terminals. Each pair is composed of a sender terminal (Sun Ultra 60, 360 MHz × 2 CPU, 512 MB memory, Solaris 2.6) and a receiver terminal (Sun Ultra 10, 333 MHz CPU, 128 MB memory, Solaris 2.6). The sender and receiver terminals are accommodated by the routers (Cisco 2500) through a 10BASE-T Ethernet. Two routers are connected by a serial line via a data channel simulator (ADTECH SX/14) which generates a reference clock for the serial line to support any line speed and inserts a propagation delay between routers.

The test conditions are as follows.

- For the duration of more than 150 sec, real time H.261 encoded video data are transferred from the sender to the receiver.
- The video data transfer is performed in a

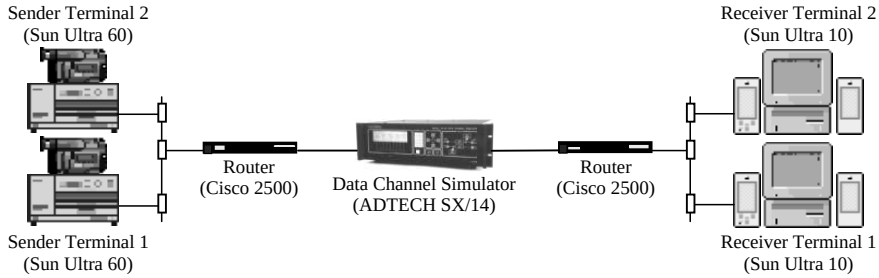


Fig. 7 Network configuration for evaluation.

pair of sender and receiver terminals (i.e., one video transmission), or is performed concurrently in two pairs of terminals (i.e., two video transmissions) using the same video source.

- In order to realize a bottleneck link, the line speed between routers is set to 1 Mbps (= 1000 kbps) for both one and two video transmissions, or 2 Mbps for two video transmissions.
- The propagation delay inserted between routers is 0 or 100 msec.
- Each DT has 512 byte packet length.
- The sender video coding layer uses one of the following H.261 coding parameters indicated in the **Table 1**. In all cases, the value of maximum quantization is set to the highest value (= 31) in order to keep the frame rate high as possible.
- The sender data transfer layer discards a video frame from the segmenting buffer when the frame has been stored there for more than 500 msec.
- As for the protocol parameter, we used the following values:
  - $\alpha = 10$ msec, $\beta = 100$ msec and $\varepsilon = 0.1$ for detecting network congestion,
  - $A = 1.1$ and $B = 2$ as increasing ratios of *int*,
  - $a = 16$ pps (packets per second) and $b = 0.5$ as decreasing parameters of *int*,
  - $\Delta = 4$ sec and $\delta = 0.8$ for adjusting coding rate, and
  - $\mu = 2$ to reflect the total amount of DTs transmitted under the present coding rate in calculating the next coding rate.

Figures 8 through 13 show the results observed on sender terminals. These figures give the following values.

- **coding rate**: coding rate of video frames requesting to the SunVideo card.
- **calculated DT rate**: DT transmission rate calculated from the value of *int*.
- **actual DT rate**: DT transmission rate per video frame being actually sent out.

Table 1 Variety of H.261 coding parameters.

coding rate [kbps]	frame rate [fps]	maximum quantization
1536	15	31
1024	15	31
768	15	31
512	15	31
384	15	31
256	15	31
128	7	31
64	7	31

6. Discussions

(1) **Figures 8** and **9** depict the results when only one video transmission is passing through the 1 Mbps bottleneck serial line. Although the calculated or the actual DT rate fluctuates frequently, the coding rate stays near the 1 Mbps serial line rate (i.e., 1024 or 768 kbps) after about 10 seconds irrespective of the presence of the propagation delay. According to these results, it is considered that our protocol can estimate the bandwidth of the bottleneck link properly.

(2) **Figures 10** and **11** depict the results for each sender terminal when two video transmissions are multiplexed on the 1 Mbps bottleneck serial line, and **Figs. 12** and **13** show the results in the case of the 2 Mbps bottleneck serial line. Independent of the value of the propagation delay, the major part of the coding rate results in nearly half of the serial line rate, i.e., 512 or 384 kbps for the 1 Mbps line rate and 1024 kbps for the 2 Mbps line rate.

Additionally, we also evaluated three video transmissions through 1 Mbps bottleneck serial line in order to observe behavior of our protocol

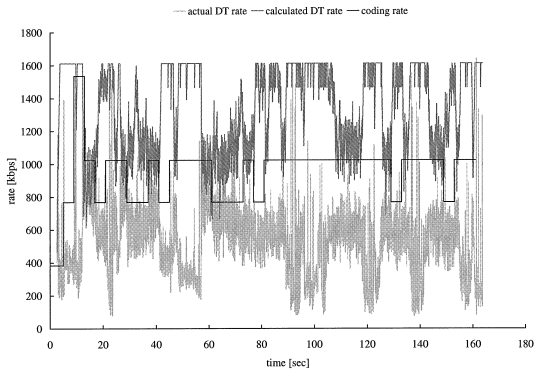


Fig. 8 Results for one video transmission (through 1 Mbps line, no delay).

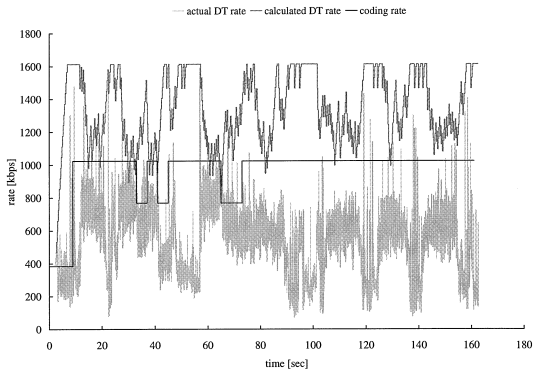
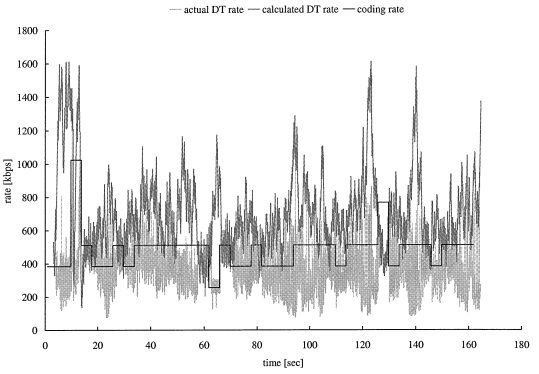
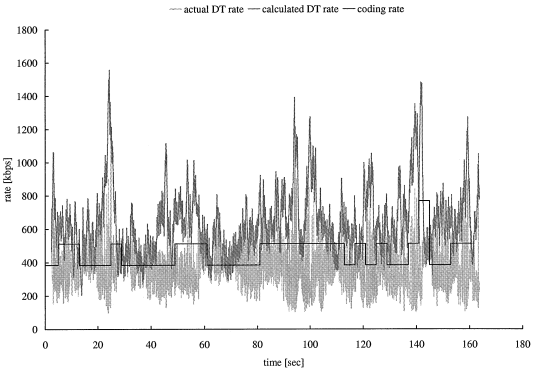


Fig. 9 Results for one video transmission (through 1 Mbps line, 100 msec delay).



(a) Sender Terminal 1



(b) Sender Terminal 2

Fig. 10 Results for two video transmissions (through 1 Mbps line, no delay).

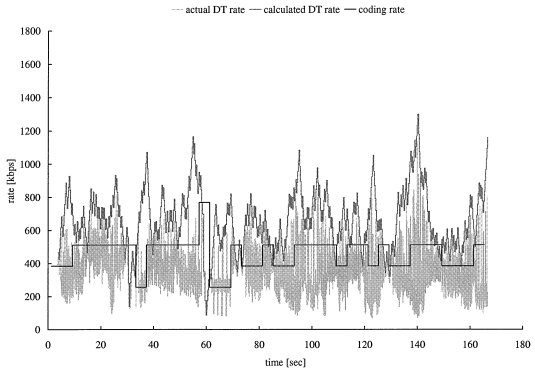
on the configuration with more video transmissions. The results are shown in **Fig. 14**. Our protocol keeps the coding rate within the range between 128 kbps and 512 kbps, where the expected coding rate is considered to be 256 kbps or 384 kbps.

Therefore, it is considered that our protocol is able to share the bandwidth of the bottleneck link fairly with the same protocol in multiple terminals.

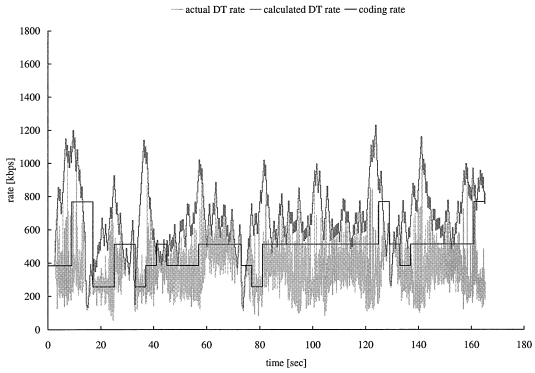
(3) Generally, it is possible that the actual DT rate is lower than the indicated coding rate because of the characteristics of video data source or those of video encoder. In our evaluation, such trends are observered in the cases that the coding rates are 1024 kbps (see Figs. 8, 9, 12, 13). In such cases, the algorithm with μ described in Section 3.3 (4) will be effective. For example, at around the time of 50 sec in the video data, the actual DT rate is much lower than the coding rate (see Figs. 8, 9). In this situation, the calculated DT rate is almost its maximum value. Therefore, the rate corresponding to int_k calculated according to Section 3.3 (2) becomes higher than 1024 kbps. However, Section 3.3 (4) will keep the coding rate at 1024 kbps because the actual DT rate is low.

On the other hand, the actual DT rate is close to the coding rate in the cases that the coding rates are 512 kbps (see Figs. 8, 9). In such cases, the next coding rate will be determined according to Section 3.3 (2). These results show that our protocol can determine the appropriate coding rate even if the sender has not transmitted DTs according to the value of *int* determined in the data transfer layer.

(4) As described in previous sections, the features of our video transfer protocol (and our previous one) are the use of two level rate control, one in the data transfer level and the other in the video coding level. As a result, it is possible that the data transfer rate is changing frequently, like TCP traffic, in order to detect the conditions of network congestion, and that the video coding rate is determined corresponding to an average data transfer rate. On the

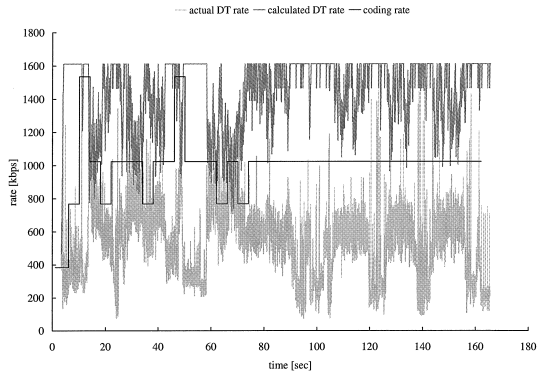


(a) Sender Terminal 1

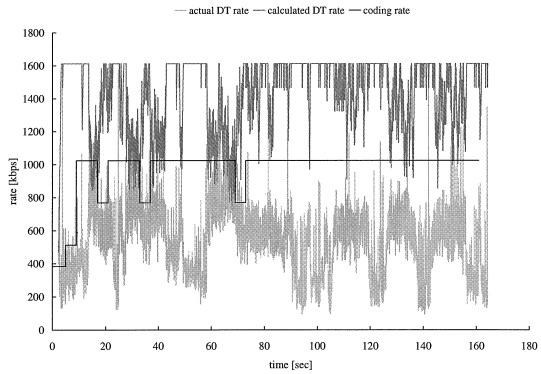


(b) Sender Terminal 2

Fig. 11 Results for two video transmissions (through 1 Mbps line, 100 msec delay).



(a) Sender Terminal 1



(b) Sender Terminal 2

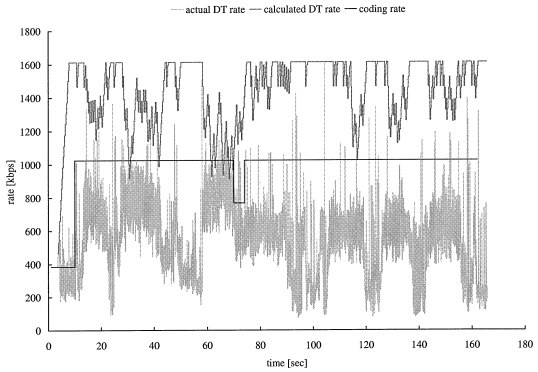
Fig. 12 Results for two video transmissions (through 2 Mbps line, no delay).

other hand, the conventional scheme with one level rate control causes an oscillation problem when multiple flows of video traffic are multiplexed into a bottleneck link. In order to clarify these situations, we have performed a simulation study on the conventional method proposed in Ref. 1). In this study, two flows of video transfer with 10 different coding rate of 1000 kbps through 100 kbps with 100 kbps steps are multiplexed in a 1 Mbps link with 10 msec propagation delay. For the simplicity, we assumed that the original video has enough information for 1 Mbps coding rate. **Figure 15** shows the change of coding rate of one video traffic. It repeats increasing and decreasing video coding rate in the range of 300 kbps through 600 kbps. This is the oscillation problem we mentioned and the comparison between Figs. 10 and 15 can conclude that our approach can avoid this problem.

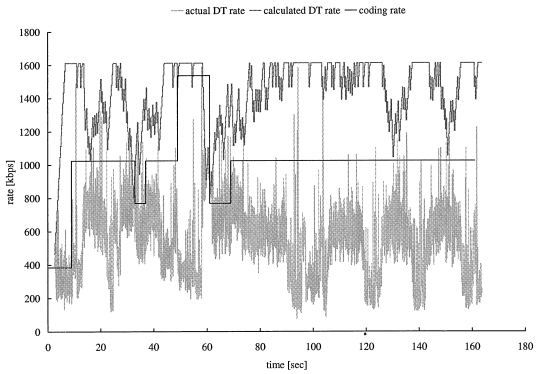
(5) This paper proposes the modified protocol for our previous method described in Ref. 6). The advantages of this paper over the previous one can be summarized in the following way.

- In the previous paper, we used only the simulation in order to evaluate the performance of the proposed protocol. On the other hand, this paper shows the results of actual implementation of a video transfer system. That is, it is considered that this paper has shown that our proposed protocol works well in actual video communications.
- As described in Section 1, our previous protocol had some overhead problems, such that ACK packets are returned for individual DT packets and that DT packets are transmitted in a specific rate, even if there are no video data to be transferred. Our new protocol described in this paper has solved those problems.

As for the second advantage, we can discuss its improvement as follows. In the results shown in Fig. 8, the sender transmitted 21658 DTs and received 4259 ACKs. In Fig. 10, 15061 DTs and 3365 ACKs were exchanged. If our pre-

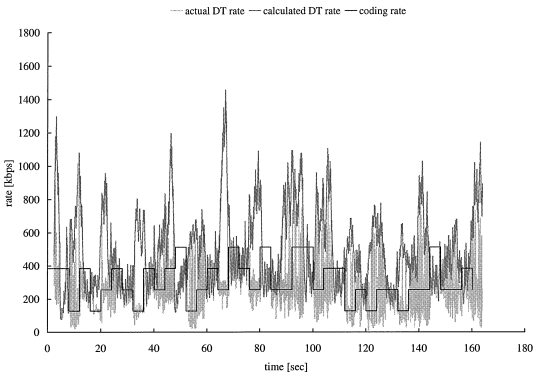


(a) Sender Terminal 1

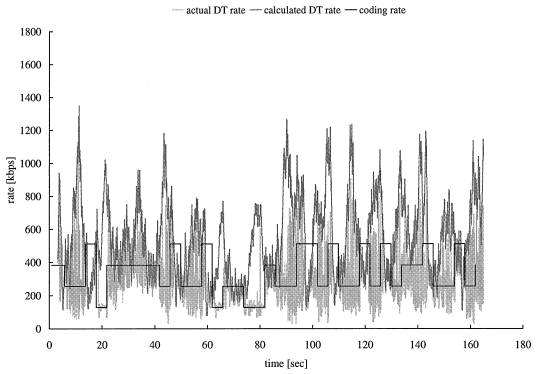


(b) Sender Terminal 2

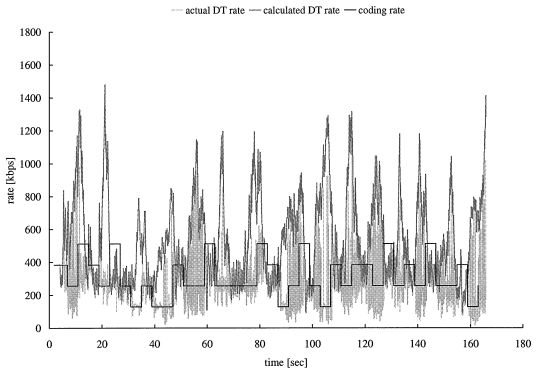
Fig. 13 Results for two video transmissions (through 2 Mbps line, 100 msec delay).



(a) Sender Terminal 1



(b) Sender Terminal 2



(c) Sender Terminal 3

Fig. 14 Results for three video transmissions (through 1 Mbps line, no delay).

vious protocol is used, the number of ACKs is equal to that of DTs. These results show that our new protocol can suppress the number of ACKs to less than 1/4 of our previous protocol. Furthermore, our new protocol does not always transmit DTs in the calculated DT rate. That is, in Figs. 8 through 14 the actual DT rate is lower than the calculated DT rate. This comes from the fact that the video source does not have enough information for the calculated DT rate. If our previous protocol is used, unnecessary DTs are transmitted at the calculated DT rate. This means that our new protocol avoids the waste of network bandwidth largely.

(6) Our evaluation mainly focuses on whether our protocol can estimate the QoS properly and can share a network fairly. In order to verify them, it is required to observe the sender side behavior where the protocol estimates the QoS and decides both the coding rate and the DT transmission rate. However, it is also important to evaluate how DTs get to the receiver because this affects the video reproduction quality. For this purpose, we obsereved the variation of the average one-way delay reported from the receiver. In the results shown in Fig. 8, the maximum value of the average one-way delay was 69 msec, and the average value was 13 msec. In Fig. 10, these values were 240 msec and 18 msec. According to these results, it is considered that our protocol can work without adding untorelable delay variations.

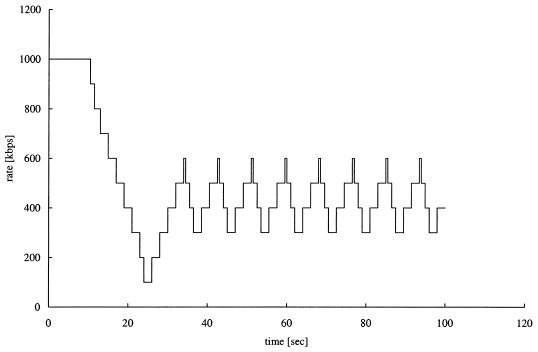


Fig. 15 Results of a method based on Ref. 1) for two video transmissions (10 msec delay).

7. Conclusions

In this paper, we proposed a video transfer protocol with congestion control using two levels of rate control based on the previous protocol proposed in Ref. 6). The new protocol improves the following items in order to apply to real video communication.

- By use of an average one-way delay to estimate QoS, the number of ACKs, which used to be the same as the number of DTs in the previous protocol, is largely decreased. Also, this prevents the estimated QoS from being affected by the situation in the opposite direction to DT transmission.
- Some problems caused by the rate mismatch between the video coding layer and the data transfer layer are solved. The sender never transmits unnecessary DTs in order to check whether there is no network congestion. It is possible to avoid inserting a large buffering delay when the coding rate is much higher than the DT transmission rate.

This paper also described the implementation of a video transfer system using our improved protocol and a commercial video tool with H.261 coding. The results of evaluation show that our protocol can estimate the available bandwidth of the network properly and share the bottleneck link fairly between two video transmissions regardless of the difference between the coding rate and the actual DT rate and regardless of the propagation delay.

Acknowledgments The authors wish to thank Dr. S. Akiba, President & CEO of KDDI R&D Laboratories Inc., for the continuous encouragement of this study.

References

1) Sakatani, T.: Congestion Avoidance for Video over IP Networks, *Proc. International COST 237 Workshop: Multimedia Transport and Teleservices* (1994).
2) Kanakia, H., Mishra, P. and Reibman, A.: An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport, *Proc. ACM SIGCOMM '93*, pp.20–31 (1993).
3) Bolot, J. and Turletti, T.: A Rate Control Mechanism for Packet Video In the Internet, *Proc. INFOCOM '94*, pp.1216–1223 (1994).
4) Mahdavi, J. and Floyd, S.: TCP-Friendly Unicast Rate-Based Flow Control, Draft posted on end2end mailng list (1997).
5) Turletti, T., Parisis, S. and Bolot, J.: Experiments with a Layered Transmission Scheme over the Internet, *INRIA Rapport de recherche*, No.3296 (1997).
6) Kato, T., Hasegawa, T., Kimura, A. and Suzuki, K.: A Continuous Media Transfer Protocol with Congestion Control Using Two Level Rate Control, *IEICE Trans. Comm.*, Vol.E82-B, No.6, pp.827–833 (1999).
7) ITU-T: Video Codec for Audiovisual Services at $p\times64$ kbit/s, Recommendation H.261 (1993).
8) Hasegawa, T., Hasegawa, T. and Kato, T.: Implementation of Video Transfer System with Congestion Control based on Two Level Rate Control, *IPSJ Multimedia Communication and Distributed Processing System Workshop*, Vol.99, No.18, pp.159–164 (1999).
9) Hasegawa, T., Hasegawa, T. and Kato, T.: Implementation and Evaluation of Video Transfer System over Internet with Congestion Control based on Two Level Rate Control, *Proc. 6th International Conference on Real-Time Computing Systems and Applications* (*RTCSA '99*), pp.141–148 (1999).
10) Stevens, W.: *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, MA (1994).
11) SunSoft: XIL Programming Guide (1997).
12) Sun Microsystems: SunVideo Plus for PCI User's Guide (1998).

(Received March 29, 2000)
(Accepted March 9, 2001)

Editor's Recommendation

This paper discusses a protocol for efficiently transmitting video data in the Internet. In this paper, the authors newly propose a scheme for resolving congestions in networks by supporting two levels of rate control. In addition, the protocol discussed in this paper are implemented and evaluated. The results obtained by this

paper are very useful to realize multimedia applications in the Internet.

(Chairman of SIGDPS Makoto Takizawa)

Teruyuki Hasegawa received the B.E. and M.E. degrees of electrical engineering from Kyoto University, Japan, in 1990 and 1992, respectively. Since joining KDD in 1992, he has been working in the field of high speed communication protocol and ATM. He is currently a research engineer of Network Management System Lab. in KDDI R&D Laboratories Inc.. He received Best Paper Award for Young Researchers of the National Convention of IPSJ in 1996, Best Paper Award of the National Convention of IPSJ in 1999, and Young Engineer Award of IEICE in 1999.

Toru Hasegawa received the B.E. and M.E. degrees in information engineering from Kyoto University, Japan, in 1982 and 1984, respectively. He received Dr. Informatics degree from the graduate school of informatics, Kyoto University in 2000. Since joining KDD in 1984, he has been working in the field of formal description technique (FDT) of communication protocols. From 1990 to 1991, he was a visiting researcher at Columbia University. His current interests are high-speed protocol and ATM. He is currently the Senior Project Manager of Network Management System Lab. in KDDI R&D Laboratories Inc.. He received IPSJ Convention Award in 1987.

Toshihiko Kato received the B.E., M.E. and Dr. Eng. degrees of electrical engineering from the University of Tokyo, in 1978, 1980 and 1983, respectively. Since joining KDD in 1983, he has been working in the field of formal specification, implementation, conformance testing of communication protocols, distributed systems, ATM and high-speed protocols. From 1987 and 1988, he was a visiting researcher at Carnegie Melon University. He is currently the Senior Manager of Mobile IP Network Lab. in KDDI R&D Laboratories Inc.. Since 1993, he has been a Guest Associate Professor of Graduate School of Information Systems, the University of Electro-Communications. He received Motooka Award in 1990.