ON REALTIME REMOTE DISPLAY OF A DIGITAL VIDEO RECORDING SYSTEM

Jenq-Neng Hwang and Qiang Liu

Dept. of Electrical Engineering, Box #352500
University of Washington, Seattle, WA 98195
{hwang, liuq}@ee.washington.edu

ABSTRACT

Digital Video Recording (DVR) systems have been popular these years following the rapid development of digital video devices and video coding techniques. A DVR is a computerized surveillance system that supports all the features of traditional videotape system by using digital cameras, motion detector, pan & tilt controls, capture cards and a computer. It can save the captured frames as harddisk files, which can be playbacked later. Another advantage of DVR is that it supports real-time remote display of the captured video frames while the digital recording is working at the same time. In this paper, we present the research on the real-time remote display of a DVR system, especially the congestion control algorithm for the multi-channel transmission case. Our proposed congestion control algorithm can achieve efficient and stable real-time transmission of multi-channel captured video sequences for a wide range of bandwidth capacity. The algorithm is also TCP-friendly as it follows the Additive Increase Multiplicative Decrease (AIMD) scheme. Several trade-offs are made to fit for the special purpose of remote surveillance.

1. INTRODUCTION

A DVR is a computerized surveillance system that supports all the features of traditional videotape system by using digital cameras, motion detector, pan & tilt controls, capture cards and a computer. It can save the captured frames as harddisk files, which can be playbacked later. A DVR is considered more convenient and reliable than the traditional time lapsed videotape system and its usage has been popular these days.

A DVR has another advantage over traditional videotape system in that it supports real-time remote display of the captured video frames while the digital recording is working at the same time. For example, a factory owner can install several cameras on site and setup a DVR system for recording. The owner can then use a client software connecting to the DVR server and receive the captured video in real-time, no matter where he or she is.

From the point of view of network transmission, the format of the client software is rather irrelevant: it can be a plug-in within a browser, or a standalone appliance. The only requirement for the client part is that it can send some performance statistics to the server. UDP transport protocol has been used to provide the real-time deliveries of video frames. As a consequence, packet loss may happen and cause quality degradation to the current video frame and possibly the following video frames (if inter-frame dependencies exist, such as the use of P frames). The client can request a key frame refresh, which implies that the video codec is required to support starting a key frame encoding at any time point.

When the client has limited bandwidth, it has to request the server to send video at a lower rate. This may impair the quality of the recorded video files on harddisk. A scalable video codec can be used so that it can save full-quality video on harddisk while send adaptive rate video into the network. We assume the DVR server has applied the scalable video coding technique and only focus on the network rate adaptation part. Typically, a DVR system has multiple cameras (4 cameras and 16 cameras are most popular, as shown in Fig. 1). A client can request to receive or not receive any of them. Also, a client can set the desired video quality for each camera independently. This implies that the DVR server needs a rate allocation scheme to assign an appropriate target rate for each active camera (an active camera means at least one client requests to display it) from the total available bandwidth.

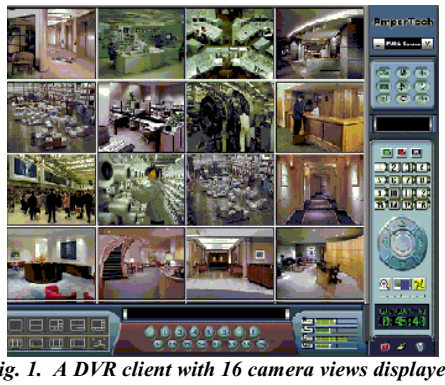

Fig. 1. A DVR client with 16 camera views displayed.

Since the end-to-end available bandwidth from the server to the client is changing all the time due to all other competing usage of the network, the received video quality may also change along time accordingly. This instability is not desirable for the display of video sequence. For remote surveillance purpose, stability issue should be addressed. A tradeoff between stability and efficiency has to be made. With limited bandwidth, a client also needs to make a trade-off between quality and frame rate. Higher quality means larger frame size and thus lower frame rate. Normally, it is necessary to have higher frame rate for the video with more motions.

Another issue is the show up speed of the initial pictures. Although the performance in the steady phase (after a while since the connection is established) is more important, the users always expect to see all the videos as soon as possible after the connection is established. The fast show-up of initial pictures

might not be possible if the user also sets the video to high quality because the initial key frames (I frame) may take a long time to arrive the client. This issue can be solved by one more trade-off at the startup phase. As the bandwidth is always shared with multiple users in the Internet, it is necessary for the transmission scheme to be friendly to other connections. Since a large body of the network traffic uses TCP, the adopted protocol should be TCP-friendly.

In this paper, we present a network transmission scheme that can support the real-time remote surveillance of DVR system. The scheme addresses all the fore-mentioned issues. It uses multi-channel token-bucket model to allocate bandwidth to different cameras; it achieve stability by using adaptive parameters in the congestion control algorithm; it is TCP-friendly by using Additive Increase Multiplicative Decrease (AIMD) rate adaptation. The rest of this paper is organized as follows. We review some of the related work in Section 2. In Section 3, we present various aspects of the adaptive transmission scheme. Descriptions of our experiment results are presented in Section 4. Finally, Section 5 concludes the paper and discusses some of our future work.

2. RELATED WORK

There has been a large body of work on congestion control for rate-based UDP applications. Rejaie, et al. proposed RAP [1], where the sender controls the rate by using packet-loss and round-trip-time information to ensure TCP-friendliness. Sisalem and Schulzrinne proposed LDA, which is also a sender based adaptation scheme [2]. Packet loss and round-trip-time are feedbacked through RTP. LDA applies dynamic determination of the Additive Increase Rate (AIR).

Packet-pair technique [3-5] can be used to estimate the bottleneck bandwidth with as few as just two packets. Without cross traffic, the dispersion of arrival time of two back-to-back packets reflects the path bottleneck. The packet-pair model assumes that the two packets queue together at the bottleneck link and at no later link. Jain and Dovrolis used *Pathload* [6] to estimate end-to-end available bandwidth. Their method is based on detecting the one-way delay increase trend, which is an indicator of congestion. Liu and Hwang extended their work to allow the usage of packets with variable size, which can be the video data [7].

MPEG-4 fine-grain scalable (FGS) video codec [8] has made it very simple and flexible for the video sender to adapt to the network dynamics. FGS is also packet-loss resilient.

Our study differs from pervious studies in that it supports correlated multi-channel videos and the stability of the algorithm is explicitly addressed

3. THE ADAPTIVE TRANSMISISON SHCEME

3.1. TCP Connection: A Control Channel

We assume there is a TCP connection between the client and the server besides the UDP "connection" from the server to the client. The authentication can be conducted through this reliable TCP connection. Also the TCP connection can be used to transmit control signals such as (1) Packet loss indicator (including which camera has loss), (2) One-way delay trend notification, and (3) Bottleneck bandwidth estimation notification. All these control signals are sent from the client to the server.

3.2. Video Frame Packetization

The proposed transmission scheme doesn't require any specific video codec. The only requirement is that the codec can refresh a key frame at any time point. We allow the user to choose what level of quality for each camera. The server then adapts to the available bandwidth by changing the frame rate. With a higher quality setting, the camera will have a lower frame rate compared to other cameras.

Depending on the quality setting of the camera, the encoded video frame varies in size. For a 320×240 size video, the encoded key frame size using the highest quality may be more than 20K bytes. Packetization is required for such a large video frame to be safely transmitted through the Internet. In our experiment, we use fixed packet size of 1400 bytes, although the user can configure this setting.

There are some protocol parameters written into the packet header including packet number (independently counted per camera; used for packet loss detection), sending timestamp (used for one-way delay calculation) and camera number. The client uses these parameters to detect packet loss, calculate one-way delay and for other purposes.

3.3. Bottleneck Bandwidth Estimation

The server periodically sends packet pairs of random size (between 1000 to 1400 bytes) to the client. After the client receives the packet pairs, it can retrieve the sending timestamp from the packet header and calculate relative delay of the two packets. An estimation of bottleneck bandwidth is obtained by using the method proposed in [5]. The client then notifies the server this bottleneck bandwidth estimation though the TCP control connection. The server calculates and keeps both median value and maximum value of these bottleneck bandwidth estimations reported from the client. Note the median value and maximum value are updated accumulatively whenever a new estimation arrives.

The median value and maximum value of the bottleneck bandwidth estimation are used in the congestion control to adjust the target sending rate. Because the bottleneck bandwidth between the server and the client is unlikely to change except for some abnormal cases (such as a route change along the path), the period of the packet pair sending is set to 1 minute in our experiments. However, to speed up the bottleneck bandwidth discovery, the period is set to 1 second within the first 30 seconds after the client connects to the server.

3.4. One-way Delay Trend

The client can use the local receiving timestamp and the sending timestamp contained in the packet header to calculate the one-way delay of each packet. For every fixed number (typically, 50) of packets, the client can calculate whether there is an increase trend among the one-way delays of these packets. The trend can be calculated as in [7] and the result is sent to the server if an increase trend exists. The server can use this information to help adjust the target sending rate. Specially, if an increase trend

exists for the one-way delay, it implies the current sending rate has reached the end-to-end available bandwidth. The server stops increasing target sending rate for a while after it receives an increase trend notification from the client.

3.5. Target Sending Rate Adaptation

The server maintains a target sending rate, which is the target total sending rate for all the cameras, including the packet header, IP/UDP header, and packet-pair overhead. The server tries to send the encoded video conforming to this target sending rate. The server also monitors and maintains the actual sending rate of all cameras. It adapts the target rate by using the statistics reported from the client. Simply put, it increases the target rate when there is no congestion while it decreases the target rate when congestion occurs. The algorithm follows AIMD strategy to be TCP-friendly. However, considering the issue of stability, we use 60% to 80% for the decreasing rate (TCP uses fixed 50%). The algorithm decreases the target sending rate when one of the following events happens: (1) At least one camera has packet loss of 5% or more, or (2) half or more cameras of the all the *active* cameras have packet loss. The algorithm is more conservative at decreasing rate than TCP to help the stability of the target rate. Table 1 shows how the target rate is decreased:

	Actual rate at the same level with target rate	Actual rate is much small than target rate
Rate decreased recently	80% of target rate	80 % of actual rate
Rate not decreased recently	60% of target rate	60 % of actual rate

Table 1. How the target sending rate is decreased

At the same time of decreasing the target sending rate, all cameras with packet loss are forced to refresh a key frame. The server also records the target sending rate when the last packet loss happens. This *target-rate-at-loss* is used to adjust the behavior of the rate increasing. When there is no packet loss recently, the server tries to increase the target sending rate. However, to improve the stability of the rate adaptation and offset the conservative rate decreasing, the rate increasing is also more conservative than TCP. The target rate is attempted to increase every 3-5 seconds with the exceptions of: (1) an increase trend of one-way delay is reported in the last minute; or (2) current target rate is 10% larger than the actual rate; or (3) current target rate is higher than the median value of bottleneck bandwidth estimation; or (4) current target rate is higher than 75% of the maximum value of the bottleneck bandwidth estimation.

When it is allowed to increase the target rate, 1% increase is used when (1) current target rate is lower than *target-rate-at-loss*; and (2) no packet loss happens in the last 30 seconds; and (3) current target rate is no more than 10Kbps larger than actual rate. Otherwise, 0.1% is used for the rate increase. We use here many empirical parameters, which have been tested in our experiments. The algorithm can in fact work well with a large range of parameters assignment, i.e., the algorithmic performance is not very sensitive to the choice of parameters. The basic instinct for those parameters is to achieve a more conservative target rate increase/decrease and thus a more stable video quality. The initial target sending rate at the beginning of a connection has to be configured by the user at the server.

3.6. Sending Rate Allocation

With a constantly updated target sending rate, the server is ready to allocate this total rate to each individual camera. A naïve strategy is to allocate the total rate evenly to each camera and let them to encode and send packets independently. However, by letting all cameras operate independently, the total rate is not smooth, especially at the initial stage when all cameras are trying to send the big key frames.

We use a multi-channel token bucket model [9] to control the sending rate of each individual camera. Each camera maintains a token bucket with size of 1K bytes, while all cameras share a global token bucket with size of 20K bytes. The token generating rate for the global bucket is the target sending rate while the bucket of each individual camera has a share of the target sending rate (depending on the active camera number) as its token generating rate. When a camera has a video frame ready for sending, it checks both the global bucket and the bucket of itself. The video frame is encoded and sent only if both buckets are non-empty. Some adjustments are used when the actual sending rate of a specific camera is considerably higher or lower than it normal share of the total target sending rate. By using this multi-channel token-bucket model, the total target sending rate is fairly allocated to each camera and the total actual sending rate is much smoother.

3.7. Startup Phase

The user always expects to see the initial pictures of all cameras as soon as possible after the client is connected to the server. However, at the beginning of the connection, all cameras are trying to send out the big key frames and there is considerable delay for the initial picture show up (especially when the video quality is set to high, which means bigger key frame). To speed up the initial picture show up, the following steps are taken at the startup phase:

(1) Enforce lower quality within the first 30 seconds regardless of user's setting, so the key frame is smaller.
(2) Shift the timing of key frame sending of different cameras evenly in a 2 –3 seconds spread.
(3) Refresh key frame more frequently (such as one key frame per 5-10 seconds) within the first 30 seconds in case some key frame packets are lost.

4. EXPERIMENTS AND RESULTS

We conducted extensive Internet experiments using a real DVR system implementing our congestion control algorithm. MPEG-4 FGS video codec [8] is used for the video. Various network conditions including LAN, Between ISPs, University to ISP, Across-Pacific, and even dial-up are tested. The results are satisfactory as the algorithm can achieve stable and smooth sending rate for long time period under various network conditions.

Our testing DVR system has 16 cameras. The video size is 320×240. To have an acceptable subjective quality and frame rate for all 16 cameras, 200Kbps to 500Kbps bandwidth is

required. However the algorithm also works for the dial-up case from the point of view of congestion control. The sending rate is rather smooth because the multi-channel token bucket model can regulate the rate effectively. Of course, the frame rate is very low for the dial-up case. Fig. 2 shows the calculated target sending rate and the actual rate for a time period of about 4 minutes.

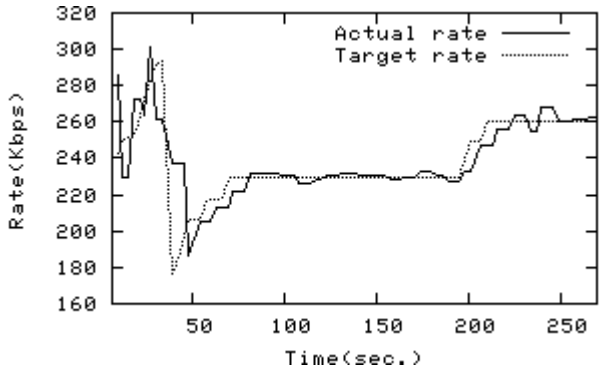

Fig. 2. Target rate and actual rate in an ADSL session.

With the startup phase speed up enhancement (Section 3.7), all cameras can show up the initial picture within 2 seconds for a 200Kbps or higher connection. Without it, the worst case for initial picture show up can be as late as 15 seconds.

Through the experiments, we have found out that all cameras tend to interleave their key frames rather evenly after the connection is established for a while, although the key frames of different cameras are sent out almost at the same time at the beginning. Fig. 3 shows that how many key frames in total are sent out each second for the beginning 3 minutes.

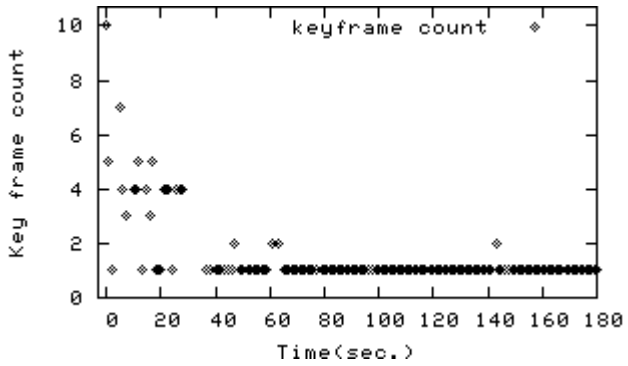

Fig. 3. Key frames refresh frequency.

5. CONCLUSIONS AND FUTURE WORK

We have presented an adaptive transmission scheme for real-time remote display of multiple cameras for a DVR system. The scheme applies the AIMD rate adaptation with explicit consideration of video quality stability. A multi-channel token bucket model is used to allocate the calculated target rate among multiple cameras. Packet-pair technique is used to estimate the bottleneck bandwidth along the path and one-way delay trend detection is used to estimate the available bandwidth. The adopted congestion control is more conservative at increasing and decreasing rate compared with TCP in order to achieve a more smooth video quality.

We also show the results of some Internet transmission experiments. MPEG-4 FGS video codec are used to encode the captured video. The results are satisfactory as the algorithm can achieve stable and smooth sending rate for long time period under various network conditions.

We plan to continue our work in several directions. Currently the total target sending rate is evenly shared by all cameras without any differentiation. However the allocation can be adjusted by considering different level of motion of each camera. For example, the camera with high motion can be biased with a little bit more share of bandwidth than other cameras. We also plan to research the scenario of multiple concurrent clients where a one-to-many transmission is necessary.

6. REFERENCES

[1]. R. Rejaie, M. Handley, D. Estrin, “RAP: An End-toend Rate-based Congestion Control Mechanism for Realtime Streams in the” in Proc. *IEEE INFOCOM 99*, vol. 3, pp. 1337-1345, 1999.

[2]. D. Sisalem and H., “The Loss-delay Based Adjustment Algorithm: A TCP-friendly adaptation” *Workshop on Network and Operating System Support for Digital Audio and Video*, July 1998.

[3]. R.L. Carter and M.E. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," *Performance Evaluation*, Vol. 27-8, pp. 297-318, Oct. 1996.

[4]. K. Lai and M. Baker, "Measuring Bandwidth", In Proceedings of *IEEE INFOCOM*, New York, NY, USA, April 1999.

[5]. C. Dovrolis, P. Ramanathan, and D. Moore, "What do Packet Dispersion Techniques Measure?", In Proceedings of *ACM SIGCOMM*, August 2001.

[6]. M. Jain and C. Dovrolis, “End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput,” *IEEE/ACM Transactions on Networking*, Vol. 11, No. 4, pp. 537-549, August 2003.

[7]. Q. Liu and J.N. Hwang, “End-to-end Available Bandwidth Estimation and Time Measurement Adjustment for Multimedia QoS”, presented at *ICME 2003*, Baltimore, MD, USA, July 2003.

[8]. H. M. Radha, M. van der Schaar, Y. Chen, “The MPEG-4 Fine-Grained Scalable Video Coding Method for Multimedia Streaming Over IP”, *IEEE Transactions on Multimedia*, Vol. 3, No. 1, March 2001.

[9]. S. Tanenbaum, *Computer Networks*, Prentice Hall, 1996.