End-to-End Frame-Rate Adaptive Streaming of Video Data

Chi-woon Fung Soung C. Liew
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
cwfung6@ie.cuhk.edu.hk, soung@ie.cuhk.edu.hk

Abstract

We propose a best-effort video delivery scheme that can be deployed on networks in use today, including Ethernet LAN's or the Internet. This scheme detects congestion by measuring the buffer occupancy and loss rate and estimate available channel bandwidth by measuring the incoming data rate at the client receiver. Frame request rate is adjusted accordingly when congestion is detected. No extra support (such as priority control or resource reservation) is required from the underlying network. The scheme is designed for the delivery of MPEG-1/MPEG-2 video streams. Various experimental settings have demonstrated the effectiveness of this scheme. *

1 Introduction

For real-time streaming of multimedia contents, such as audio and video, over the Internet, various multimedia client-server systems have been proposed. A problem faced by these systems is the heterogeneous nature of the Internet, in which some clients are connected to the server through a broadband channel, while others are connected through dial-up modems. Within a network, traffic conditions may change [1] and the sustainable bandwidth of a particular path may vary throughout the course of the day. To dynamically adjust the data rate to the available bandwidth of the connection [2], different schemes have been proposed.

In this paper, we propose and investigate a client-pulled frame-rate adaptive video retrieval scheme. The adaptive protocol resides on the application layer, making use of UDP as the underlying transport layer. The adaptive application layer can be divided into two sublayers. At the upper sublayer, the client makes explicit requests for video frames. The lower sublayer measures the data arrival, data loss rates, and the round-trip data-delivery time, allowing the upper sublayer to adjust the frame request rate to accommodate the network bandwidth and server capacity. MPEG video is assumed at the upper sublayer. By selectively skipping the requests for visually less-important frames, the data rate can be adapted to ensure optimal video quality given a bandwidth constraint.

The rest of this paper is organized as follows. the current development of Realtime Transport Protocol. Section 2 presents our adaptive probe-up and probe-down algorithms. Experiment results are presented in Section 3. Section 4 concludes this work and discusses possible applications and extensions.

2 An Adaptive Video Retrieval Algorithm

The essence of an adaptive algorithm is to adjust the requested data rate (hence the video quality) according to the rate supportable by the system. The goal is to provide a continuous flow of video display on the client side with the best possible quality.

Congestions may occur in the network or in the server. Since the system has no particular preference for dropping video packets (whether they are packets for I, P or B frames), any one of them might be discarded. In fact, it is more likely for I frames to be dropped because of their relatively larger size. From the client's point of view, packet loss means video-quality, especially when packets for I/P frames are lost, which results in decoding-error propagation. If the client can reduce the loss rate by skipping the request for less important data (i.e. B frames) first, the impact towards visual quality will be less severe. Therefore it may be better for the client to reduce the load to the server or network voluntarily by reducing the request rate.

Figure 1 shows the conceptual model for our adaptation mechanism. The client requests data at a frame

*This work was supported by an RGC Earmarked Research Grant of the Hong Kong University and Polytechnic Grant Council: CUHK 336/96E.

0-7695-0253-9/99 $10.00 © 1999 IEEE

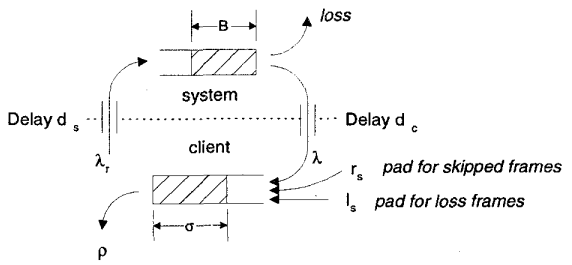

Figure 1: System Model for Lossy Environment

rate of λ_r. After some delay, this request reaches the server, upon which the server will start sending data at this rate. Both the request and data packets may get lost in transit and they may be backlogged within the network or server. As far as the client is concerned, it can treat the server and the network as a system which responds to its requests after some delay. The backlog in the system is denoted by B. Due to the possibility of loss and delay jitter, the effective data arrival frame rate λ (with corresponding data rate Λ) observed by the client at any given time might be different from the request rate λ_r (with data rate Λ_r). Artificial frames are padded to the buffer to compensate for skipped and loss frames.

The issue in adaptation is how the request rate should be adjusted based on the observation of the arrived data. Ideally, λ_r should be equal to the lesser of λ_r and λ_{upper}, where λ_{upper} is the maximum frame rate (corresponding to a data rate Λ_{upper} that can be sustained by the system. Note that the bottleneck for λ_{upper} could be either at the server or the network. The client may gauge its value only indirectly via the observed λ.

2.1 Adapting the Request Rate to the Available Bandwidth

Let us now go into the details of our adaptation scheme. At any point in time, there are three possibilities: 1) the system is working fine and no adaptation is needed; 2) the request rate is too high; 3) the request rate is too low.

Case 1: System works fine. Losses are rather random when a video stream use a small fraction of the available bandwidth. In this case, skipping requests will not help reducing the loss probability and therefore the request rate should be maintained. Studies show that a random loss rate up to 10% is possible in some network environment [9]. To absorb the effect of this random loss, a safety margin of loss rate up to ϵ of the request rate is used. When loss is detected with loss rate below the safety margin, it is assumed that the loss is not caused by system congestion and therefore request rate will not be changed.

Case2: Request rate is too high ($\Lambda_r > \Lambda_{upper}$). When the client detects data loss, it compares the measured arrival rate with the request rate effective when the request for the loss data were made (i.e., Λ_r shifted back in time). We need to consider adjusting the request rate downward only if Λ_r is more than 10% above Λ, the observed arrival rate. In many ensuing discussions in this paper, when we say $\Lambda_r > \Lambda$ and downward rate adjustment is being triggered, it should be understood that this safety margin has been taken into account. That is, if 10% is the safety margin, this means Λ_r is more than 10% larger than Λ.

Downward adjustment may also be triggered if the receiver buffer level falls below a threshold and buffer underflow may be imminent if no action is taken. A scenario under which this may happen is when there large amounts of buffers in the network or the server. For instance, the server may be congested and the requests are backlogged in the request queue in the server.

We now determine the receiver-buffer threshold σ_{min} (in unit of frames) for downward rate adjustment. At any given time, the rate at which the buffer is being drained is

$$\rho - (\lambda + r_s + l_s) = \lambda_r - (\lambda + l_s) \qquad (1)$$

Recall that B denote the outstanding request or backlog in the system. Suppose when the buffer is at threshold σ_{min}, when the client reacts by adjusting λ_r the request rate. It will take some time before this new rate takes effect since the outstanding requests must be cleared first. Assuming the outstanding request will continue to clear at the current arrival rate λ with the same loss rate l_s, it will take an amount of time $B/(\lambda + l_s)$ before the new request rate takes effect. Thus, in order to react in time to prevent buffer underflow, σ_{min} should satisfy the following inequality:

$$\frac{\sigma_{min}}{\lambda_r - (\lambda + l_s)} > \frac{B}{\lambda + l_s} \qquad (2)$$

$$\text{or,} \quad \sigma_{min} > B\left(\frac{\lambda_r}{\lambda + l_s} - 1\right) + \sigma_m \qquad (3)$$

In practice, one can add a safety margin to the above σ_{min} (say, 10 frames of video data) to set the threshold for conservative operation.

When congestion is declared under the above situations, the client would enter a *three-stage probe-down phase* to adjust the request rate downward, as described below. A flowchart is given in Fig. 2.

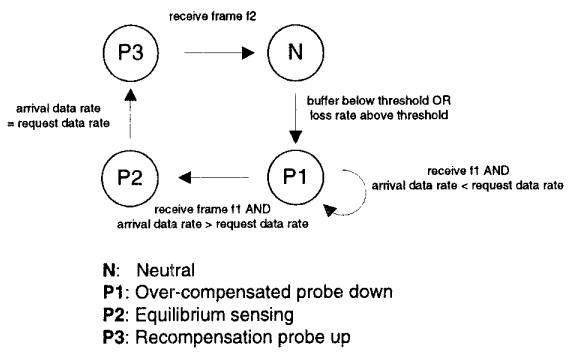

Figure 2: Three-stage probe-down phase

Three-stage Probe-down Phase

Overcompensated Probe-down Stage

The first stage of the *probe-down phase* is *Overcompensated probe-down*. The client sets Λ_r to a value such that the resulting Λ_r is smaller than Λ. By requesting at a rate lower than the supportable rate, the goal is to lower the backlog B in the system. If this works, the backlog B then gradually decreases.

With reference to Fig. 3, we mark the first requested frame after lowering Λ_r as f_1. Before the new request rate takes effect, the arrival rate should still be running at about Λ_{upper}. The client should therefore continue to measure the supportable arrival rate in the period before f_1 arrives just in case Λ_{upper} changes. The rate difference $\delta = \Lambda - \Lambda_r$ is stored for later use.

The arrival of f_1 marks the ends of the first stage. In case f_1 is lost, the next successfully received frame should mark this event. It is possible that Λ_{upper} has changed during this probe-down stage and that indeed the arrival rate is still lower than the new request rate. In this case, the probe-down phase is restarted and the overcompensation probe-down is repeated with an even lower request rate. If the arrival rate is at least as large as the new request rate, the second stage begins.

Equilibrium Sensing Stage

The second stage is the *Equilibrium sensing stage*. After the arrival of f_1 and if Λ_{upper} has not changed, the receiver buffer level σ should build up slowly (since most of the data are now pad data for skipped frames which are not transported over the network). As shown in Fig. 3, the arrival rate would gradually decrease to the current request rate. When that happens, equilibrium has been achieved and the second stage ends.

Recompensation Probe-up Stage

The third stage is the *Recompensation probe up stage*. In the first stage, the request rate was intentionally set at below the arrival rate to clear system backlog. The goal of this stage is to reset the request rate to the sustainable rate. Specifically, the client increases the request rate to the arrival rate measured previously in the first stage. At this point, the client can set λ_r by mapping the data rate $\Lambda_r + \delta$ to the frame rate, where δ is the rate difference measured before frame f_1 arrived. We mark the first request frame after the setting of λ_r as f_2. The *three-stage probing phase* ends when f_2 is received.

Of course, during the last two stages, Λ_{upper} may change and drift downward. In that case, another overall probe-down phase will be started at the conclusion of the third phase.

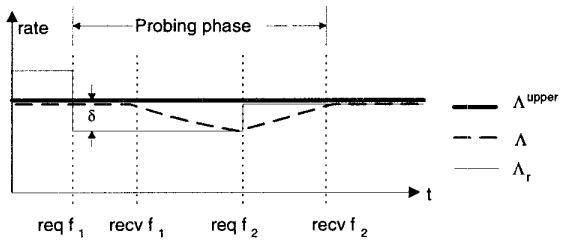

Figure 3: Probing Phase

Case 3: Request rate can be increased ($\Lambda_r < \Lambda_{upper}$). Telling when the traffic congestion has freed up and the request rate can be increase is more problematic. Unlike downward adjustment, which can be triggered by the observation that Λ is lower than Λ_r, observing Λ in this case is not enough because at best $\Lambda = \Lambda_r$ and we cannot derive the higher sustainable rate Λ_{upper} by measuring Λ. In other words, when there is insufficient bandwidth, $\Lambda = \Lambda_{upper}$, but when there is extra bandwidth $\Lambda < \Lambda_{upper}$.

Another indicator that Λ_{upper} could be higher now is that the observable RTT, the round-trip time, is smaller than before. By monitoring the changes in RTT, the possible availability of extra resources can be detected. However, because of the stochastic behaviour of the network, RTT fluctuates. To observe the long-term variation of the RTT, these fluctuations must be smoothed out first. This can be done by using a smoothing filter with a very low cut-off frequency.

In the *probe-up phase*, the client increases the request rate linearly. The value of λ_r is first increased by a small value (say, by one frame per second). This λ_r is kept until the measured arrival rate Λ matches Λ_r, the data rate corresponding to λ_r, then λ_r is increased again. This process should be repeated until a point when λ_r reaches ρ (the full rate of the unskipped

video) or when $\Lambda_r > \Lambda$, the measured arrival rate. For the latter case, the client will enter the *probe-down phase* as described in the previous section to zoom into Λ_{upper}.

3 Experiment Result and Analysis

In the experiment, we test the system in a transpacific, long distance link setting. The clients run on Pentium Pro PC with Microsoft Windows 95. The server run on a Sun Sparc 20 workstation with SunOS 5.4. To simulate the effect of system congestion, dummy clients which send extra requests are also connected to the server. These dummy clients control the system resources available to the client under test. Each dummy client makes use of UDP to inject data into the network with rate ranging from 2Mbps to 3Mbps. None of these dummy clients adapts to the network traffic condition, i.e. they transmit at a fixed rate. In case their requested rate is not supportable by the network, packets are dropped silently in the network.

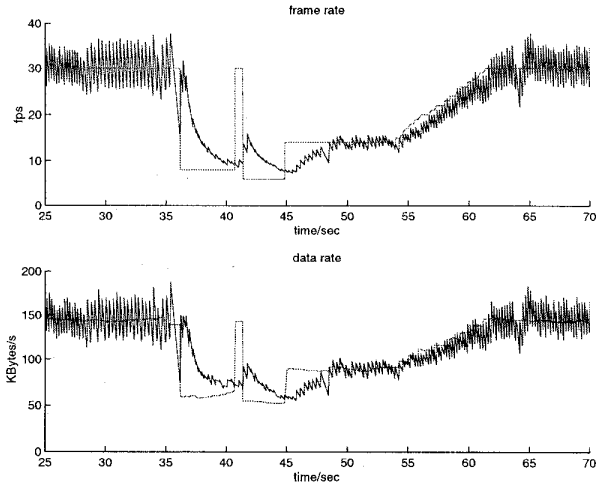

Figure 4: Snap Shot of the Adaptation Process in the Transpacific Link

we run the experiment in the transpacific link, with the available data rate of about 310KBytes/s. This data rate is measured by probing the link with UDP packets. One non-adaptive dummy client is used to generate extra traffics to create congestion. The total data rate used up by the dummy client is 220KBytes/s. The snap shot on figure 4 shows that when the dummy traffic starts their connection at time 35s, loss is detected and the request data rate is reduced. After two adaptation cycle, the request rate is stablized to 90KBytes/s at time 45s.

The dummy client stopped generating extra traffic at time 54s. A continuous drop in RTT is detected and the client entered the *probe-up* phase at time 55s. Full request rate is achieved at time 62s.

Result shows that video clients using the algorithm can adapt to the network within tens of seconds. In terms of data rate usage, the results are satisfactory. Next we will move on to the quality of the video received by the clients using the proposed algorithm.

3.1 Video Quality Assessment

For MPEG video, packet loss results in video quality degradation. When data from I or P frames are lost, subsequent P and B frames in that GOP have to be discarded. In this part we will examine the quality of the video sequences that are transmitted with and without using the algorithm in congested networks.

The same set of experiment are run twice. In the first trial, a video client that runs the adaptation algorithm competes with dummy clients that occupy the channel with fix-rated UDP traffic. The video received by this adaptive video client is recorded. In the second trial, the adaptation algorithm of the video client is turned off. It then competes with the same set of dummy clients. The video received by this non-adaptive video client is recorded. We will compare the quality of these two sets of video.

Figure 5 shows the frame rates of the output video of the two type of clients. The frame rate here means how many frames with actual data are played per second. Dummy frames stuffed because of request skipping, data loss from network and those extra loss e of the loss of data from an I or P frames are not counted in the frame rate.

The frame rate of the non-adaptive client fluctuates rather drastically. This means that although the average frame rate of the non-adaptive client is higher than that of the adaptive one, the non-adaptive video can have many sudden pauses. By contrast, although the adaptive one has a lower overall frame rate, its video quality is smoother in comparison. This overall smoother video quality has been observed subjectively in our experiment.

Three reasons account for the better video quality of the adaptive client despite the lower rate: 1) The adaptive client selectively drops less important data (i.e. B frames) outside the network where as the non-adaptive client has no choice but to let the network drops its data randomly. If I or P frames are dropped, subsequent P and B frames in the GOP cannot be decoded and thus results in a sudden pause. 2) The adaptive client drops a complete frame whereas the non-adaptive client may drop part of a frame in the network, which results in the drop of a complete frame

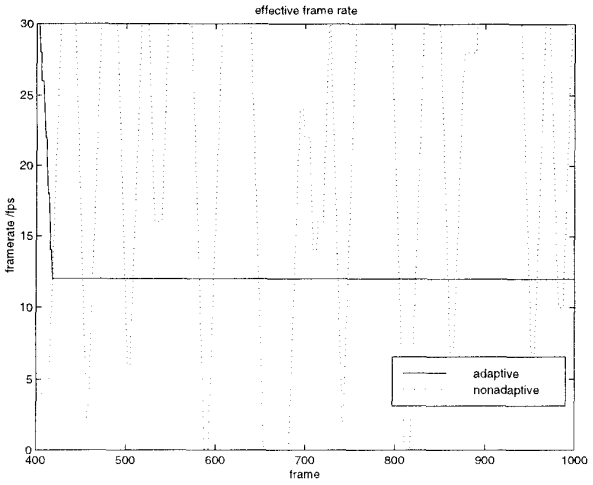

Figure 5: Frame Rate of the Resultant Video Sequences

at the UDP receiver anyway. The bandwidth used to transmitted that frame is wasted. 3) The spacing between dropped frames of adaptive clients are more even, resulting in smooth quality as indicated in fig 5.

Another observation is that the adaptive client controls the bandwidth consumption so that the total requested rate (by the adaptive client plus the dummy clients) falls below the channel capacity. Since dummy clients do not adapt to network conditions, the adaptive client reduces its share for the dummy client as well. As a result, the packet loss rate for dummy clients is lower than those dummy clients that compete with non-adaptive clients. It is interesting that when adaptive client is used, not only is its video quality better, the other non-adaptive UDP clients also receive more bandwidths.

4 Conclusions

We have proposed a client-pulled end-to-end application level frame-rate adaptation scheme for streaming video that dynamically adjusts the request rate and window size based on the measurement of arrival rate buffer occupancy and round trip time on the client side to accommodate the current network condition. Using the statistics at the client side, the client controls the data rate of the session by adjusting the request rate to retrieve a continuous flow of video from the server over the network.

Since most of the adaptation process, including statistic gathering and decision making, is done on the client side, the server is left with the job of reading and sending video data only. By off-loading the server with the adaptation process, this video delivery system can be more easily scaled up to support many clients.

Results show that using this algorithm, applications quickly react to congestion and can zoom to a suitable request rate within tens of seconds. Video continuity is preserved in most cases.

In conclusion, this algorithm is rather effective and is suitable for best effort point-to-point video-on-demand applications over channels where bandwidth is not abundant.

References

[1] A. Mukherjee, "On the Dynamics and Significance of Low Frequency Components of Internet Load", *Internetworking: Research and Experience*, 1994, Vol.5, pp 163–205.

[2] J. Gecsei, "Adaptation in Distributed Multimedia Systems," *IEEE Multimedia*, April-June 1997, pp. 58–66.

[3] S. Chakrabarti and R Wang, "Adaptive Control for Packet Video", *Proceedings of the International Conference on Multimedia Computing and Systems '94*, 1994, pp 56–62.

[4] S. C. Liew and C. Y. Tse, "A Control-Theoretic Framework for Adaptation of VBR Compressed Video for Transport over a CBR Communications Channel", *IEEE Trans. on Networking*, vol. 6, no. 1, pp. 42–55

[5] P. Pancha and Magda El Zarki, "Prioritized Transmission of Variable Bit Rate MPEG Video",

[6] S.L. Blake, S. A. Rajala, F. Gong and T. L. Mitchell, "Efficient Techniques For Two-Layer Coding of Video Sequences," *Proceedings of ICIP '94*, 1994, pp. 253–257.

[7] M. Ghanbari, "Two-layered Coding of Video Signals for VBR Networks," *IEEE J. on Selected Areas in Commun.*, vol. 7, pp. 771–781, 1989.

[8] S. A. Thomas, *IPng and TCP/IP Protocols*, New York, Wiley, 1996.

[9] Jean-Chrysostome Bolot, "Characterizing End-to-End Packet Delay and Loss in the Internet", *Journal of High-Speed networks*, September 1993, vol. 2, no. 3, pp. 305-323.

[10] V. Paxson, "End-to-End Internet Packet Dynamics", *Proceeding of SIGCOMM 97*, 1997.

[11] S.C. Liew and C.Y. Tse, "Video Aggregation: Adaptive Video Traffic for Transport over Broadband Networks by Integrating Data Compression and Statistical Multiplexing", *IEEE J. on Selected Areas in Commun.*, Vol. 14, no.6, pp 1123-1137, Aug. 1996.