A Complexity-Aware Live Streaming System with Bit Rate Adjustment

Meng-Ting Lu, Chang-Kuan Lin, Jason Yao, and Homer Chen
Graduate Institute of Communication Engineering
National Taiwan University
Taipei, Taiwan 10617, R.O.C.
b88901124@ntu.edu.tw, junglelw@gmail.com, jasonyao@cc.ee.ntu.edu.tw and homer@cc.ee.ntu.edu.tw

Abstract

For a live streaming server, it is highly desirable to allocate the available computational resource and bandwidth to each channel fairly and efficiently. In this paper, we propose a complexity-aware live streaming system with bit rate adjustment that can handle the allocation of bandwidth and computational resource of a live streaming server. The proposed system encodes the input videos at different quality levels based on the priority of the input videos and the available computational resource. It incorporates a bit rate adjustment mechanism to compensate for the video quality drop resulted from the quality level change of high priority encoders. The resulting system is able to handle channels more efficiently because the complexity of high priority encoders can also be dynamically adjusted with little quality drop. A new complexity adjustment method is developed that enables the system to stabilize more quickly and minimizes the variance of the time buffer. The experimental results show that the system can handle more channels while still maintaining the quality of high priority encoders. The proposed system is applicable to multimedia home gateways, surveillance, IP-based TV, and on-line sports game relays.

1. Introduction

Nowadays, the Internet has become the most important means of entertainment and communication. People get on the Internet to watch TV, communicate with friends by video conferencing, and rent a movie through video on demand (VoD). All these streaming applications are made possible by the establishment of the multimedia coding standards like MPEG-4 [1] and the communication protocols like RTP/RTCP [2], RTSP [3], and SDP [4]. In order to provide users with the best video quality under the resource constraint, the streaming server also needs an effective mechanism to control the streaming process. For most cases, bandwidth is the primary constraint because the Internet is a best-effort networking environment. For live streaming, however, the computational resource also becomes a critical factor. Therefore, our goal is to develop a control mechanism for the allocation of both bandwidth and computational resource on the live streaming server.

There are already some literatures on complexity-aware encoders. Tai et al. [5] proposed a computation-aware scheme that allows the searching process to stop once a specified amount of computation has been performed. Huang et al. [6] also proposed an adaptive search strategy to find the best block matching results in a computation-limited and computation-variant environment. Both schemes are based on the fact that the encoder already knows how much computational resource is allocated to them. The scenario considered in this paper is quite different. We are trying to allocate the limited computational resource and bandwidth to multiple video channels on the streaming server. The key task is to perform a trade-off between video quality, consumed bandwidth, and computational complexity for each encoder. In our previous work [7], we have proposed a complexity-aware live streaming system that only considers the allocation of computational resource. The complexity of each low priority channel is adjusted based on the current available computing power while high priority channels are always maintained at the highest quality. In this paper, we propose a new complexity-aware streaming system that integrates bit rate adjustment with our previous system to compensate for the quality drop resulted from the quality level change of high priority

Proceedings of the Seventh IEEE International Symposium on Multimedia (ISM'05)
0-7695-2489-3/05 $20.00 © 2005 IEEE

encoders. We also introduce a new complexity adjustment method that enables our system to stabilize more quickly and minimizes the variance of the time buffer. With this new scheme, our system can handle more channels without loss of video quality for high priority encoders.

The rest of this paper is organized as follows. Section 2 presents the usage scenario and system architecture of our complexity-aware live streaming system. Section 3 describes the details of our control algorithm, including the complexity adjustment, quality level allocation, and bit rate adjustment strategies. Simulation results and discussion are presented in section 4. Finally, section 5 gives a conclusion of this paper.

2. Usage scenario and system architecture

This section describes the usage scenario and system architecture of our system. The system architecture is shown in Figure 1.

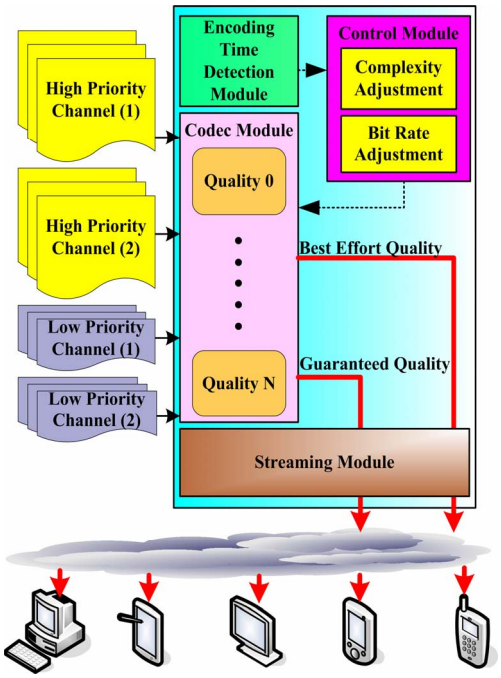

Figure 1. Complexity-aware streaming server architecture.

Assume we have two kinds of live video sources, high priority and low priority channels. It is expected that the high priority channels should be maintained at the best quality. Before encoding each frame, the control module first detects the previous encoding time of all the channels to decide how to allocate the resource for the next frame. It then calculates the amount of quality levels and bit rates to be adjusted based on the information provided by the codec module and the encoding time detection module. The result is passed to the codec module again for adjustment. With this architecture, the high priority channels are delivered with guaranteed quality while the low priority channels are served with best-effort quality. This architecture is applicable to multimedia home gateways, surveillance systems, IP-based TV, and on-line sports game relays. The codec used in our current system is XviD 1.1.0-beta2.

3. Control module

This section describes the details of the control module. As shown in Figure 2, the control module consists of three parts. The first part determines the complexity adjustment strategy. The second part decides how to allocate the shift of quality level to each encoder, and the last one chooses the bit rate adjustment strategy for quality compensation. After encoding each frame, the control module first determines the quality levels to be adjusted for all encoders. Then, the bit rate to be adjusted is decided based on their quality levels.

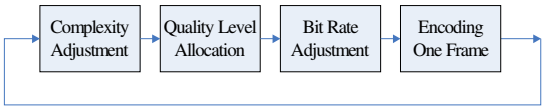

Figure 2. Control module block diagram.

3.1. Complexity adjustment strategy

In the following discussion, we define ΔQL to be the total quality levels to be shifted up or down for the next frame. We also define *TB* to be the accumulated time buffer in milliseconds, which is the accumulated difference of total actual encoding time and desired encoding time, and ΔTB to be the difference between the current *TB* and the *TB* three frames before. ΔTB is similar to the variation trend of *TB*. The proposed complexity adjustment strategy calculates the necessary total shift of quality level based on equation (1). The first term at the right side of the equation can be thought of as the component to force *TB* back to zero, while the second term compensates for the recent adjusting rate of *TB*.

$$\Delta QL = \frac{TB}{\alpha} + \frac{\Delta TB}{\beta} \quad (1)$$

In order to derive the best quality level adjustment, we have to set up several coefficient sets of α and β for different situations. The coefficient decision process is shown in Figure 3. If the result of *TB* multiplied by ΔTB is positive, the encoding time tends to deviate from zero, so we have to pull *TB* back. For instance, positive *TB* shows that the encoders have consumed too much time, and positive ΔTB indicates that *TB* tends to further increase. Therefore, we have to pull down some quality levels to meet the time constraint. The amount of total quality level shift is determined by the weighted sum of *TB* and ΔTB. Vice versa for both negative *TB* and ΔTB except that we increase the quality levels. Whenever the value of *TB* multiplied by ΔTB is negative, it means that the encoders have already been adjusted by former quality level shift so that *TB* is heading back to zero. The main issue lies in whether the adjustment is appropriate. If $|\Delta TB|$ exceed two third of $|TB|$, it indicates that the adjustment is too fast and may flip *TB* to the other side in the following few frames, so we have to slow down the rate. If $|\Delta TB|$ falls under one third of $|TB|$, it reveals that the pull back trend is not efficient enough, so we need a larger amount of quality level shift to accelerate the pull back procedure. The coefficients of α and β mentioned in Figure 3 are chosen experimentally such that *TB* will converge to zero quickly with less fluctuations.

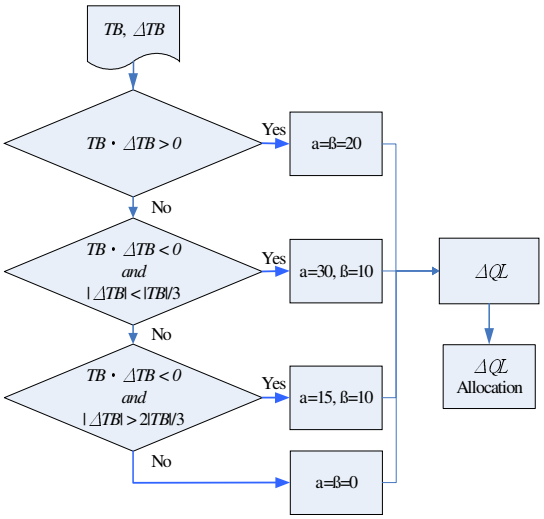

Figure 3. Flow chart of quality level adjustment.

3.2. Quality level allocation strategy

We demonstrate the overall procedures of quality level allocation in Figure 4 and explain as follows. After computing the total level shift, we next determine how to allocate the shift to each channel. For positive *TB*, we have to reduce the encoding time and thus pull down the quality level of encoders. We first allocate the quality level shift to encoders with low priority until each of their quality level reaches zero, and then the rest will be distributed among high priority encoders. The allocation within each priority group will be in a round-robin manner. In the negative *TB* case, which means the encoders have spare time after all the current frames are encoded, we may push up the quality level by the amount calculated in Figure 3. The procedure of quality level shift allocation is similar to that in the positive *TB* case, but in a reversed way. The quality levels of low priority encoders will not be raised back until all high priority encoders have reached the highest level.

After the quality level of each encoder is down to zero, the server reaches its limit and cannot decrease the encoding time anymore. To further enhance the control efficiency of *TB*, we add a new quality level, -1, which represents the skip mode. This means that the current frame will not be encoded and the encoder will use the previous frame to replace the current one. However, the new quality level should be used carefully because it may result in a significant PSNR drop. Therefore, we only apply the skip mode on low priority encoders. A cool down interval between dropped frames is essential because consecutive frame drops will cause a serious error propagation problem. The allocation of quality level -1 is different from that of quality level 0 to 6. In the case of positive *TB*, when all encoders reach quality level 0 and there is still quality level shift left, we randomly select up to two low priority encoders to assign *QL* -1 to them. All encoders of *QL* -1 will be reset to *QL* 0 after skip one frame and cannot be set to *QL* -1 in the next three frames. By doing so, we could maintain a cool down interval and avoid consecutive frame drops for the same encoder. The total quality level shift would be ignored anyway while every encoder is in the lowest quality level or in the cool down interval.

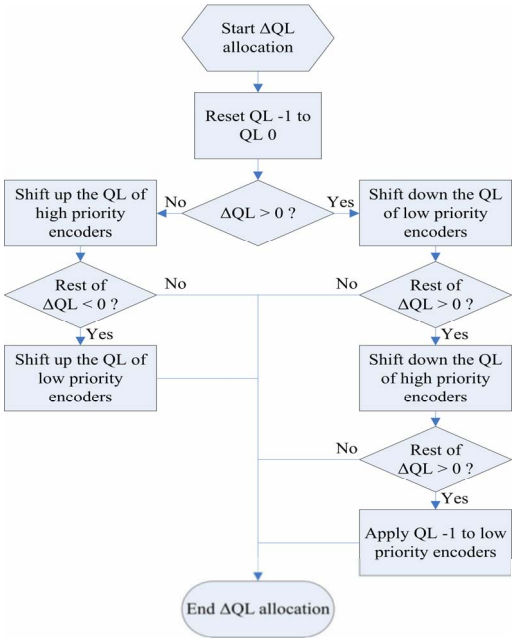

Figure 4. Flow chart of quality level allocation.

3.3. Bit Rate adjustment strategy

The complexity adjustment strategy mentioned above makes a trade-off between video quality and encoding time. However, quality drop for high priority encoders is inevitable. Therefore, we use the bit rate adjustment strategy described below to ensure the quality of high priority channels. It is obvious that raising the bit rate could improve the video quality, but the encoding time is also increased. According to our experiments, the encoding time depends on the streaming bit rate slightly, but it is greatly affected by the quality level. In Figure 5, we plot encoding time versus bit rate varying from 220 to 1,500 kbps for seven quality levels. It is obvious that the encoding time of each quality level increases slightly while raising the bit rate, but there are significant drops of encoding time while decreasing the quality level. The encoding time of higher bit rate and lower quality level is still less than that of lower bit rate and higher quality level. Thus the overall encoding time is still reduced. For example, the encoding time of 960 kbps and quality level 5 is less than that of 660 kbps and quality level 6, and we can still achieve 22% reduction of time, about 3 milliseconds. This indicates that compensating for the quality drops with higher bit rate is feasible and does not conflict with the time-saving purpose.

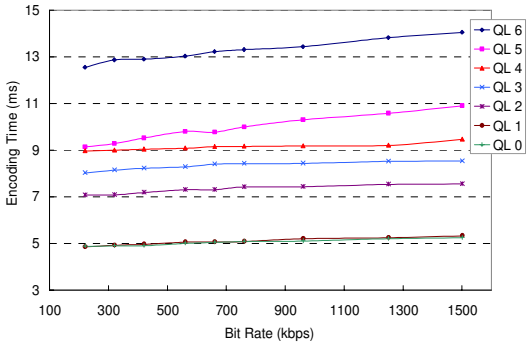

Figure 5. Encoding time versus stream bit rate.

The flowchart of the bit rate adjustment strategy is shown in Figure 6. The basic concept of this bit rate adjustment strategy is to compensate for the quality drop with higher encoding bit rate. Note that we only apply the compensation on high priority channels. Because of the premise that the total outgoing bandwidth is bounded, we sustain the higher bit rate of high priority channels by pressing down the bit rate of low priority channels. Based on the rate-distortion curves of different quality levels [7], higher quality level results in better PSNR performance. However, the rate-distortion curves of XviD MPEG-4 codec divide into two groups: quality level 2 to 6 and quality level 0 to 1. There are slight differences among each group, but the gap between groups is significant. Thus, we should give corresponding compensative weight (CW) to different quality level.

Due to the limitation of the employed XviD MPEG-4 encoder, we cannot change the bit rate too often, or the encoding bit rate may fluctuate a lot and result in imprecise rate control. Thus, we adapt the bit rate per 30 frames and use the average of the compensative weight of the previous 30 frames as the bit rate input of the next 30 frames. As the complexity adjustment strategy mentioned in section 3.1, the designated quality level for each high priority user is uncorrelated, so there should be different bit rate adjustment between them. After determining the bit rate compensation of prior users, we distribute the left bandwidth equally to the low priority users. We set a lower bound of a low priority streaming bit rate to ensure the base quality. When the bit rate of low priority reaches the boundary, we retrench the extra compensative bit rate to meet the total bandwidth limitation.

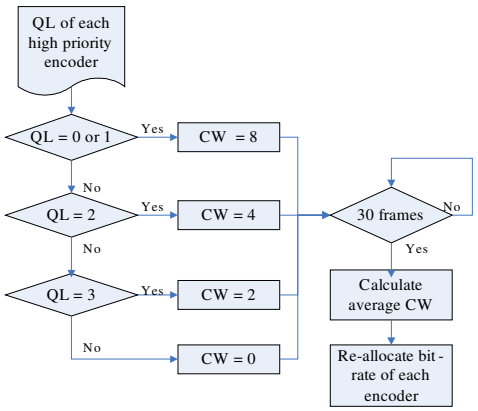

Figure 6. Flow chart of bit rate adjustment.

4. Simulation Results

In this section, the simulation results are presented. We did all the experiments on a computer with Pentium (R) 4 CPU 3.2 GHz and 768 MB RAM. All the test sequences used were at CIF resolution. All test sequences were encoded with only I and P frames with I frame interval being 30 frames. The total available bandwidth was fixed at 3 Mbps. The following experiments of complexity and bit rate adjustment consisted of two high priority and four low priority encoders. The high priority encoders were initially defined at quality level 6, bit-rate 660kbps. The low priority ones were defined at quality level 3, bit-rate 420kbps. For clarity, we only illustrate one encoder of each priority group because the results of other encoders are similar to the representatives shown. We did two experiments: the first one aims to analyse the statistics of the control results, and the second one is to test the accuracy of complexity control.

In the first experiment, we encoded the sequence Football with target encoding time 45 milliseconds and 33 milliseconds. We show the quality levels of each frame decided by the complexity adjustment strategy in Figure 7 and 11, and the resulted time buffer in Figure 8 and 12 respectively. In the case of target encoding time 45 milliseconds, quality level -1 is rarely utilized because the time constraint is loose, and the time buffer seems no different whether we enable the function of frame dropping or not. Therefore, we only show the time buffer with frame dropping function enabled in Figure 8. In the case of encoding time 33 milliseconds, we can see from Figure 12 that the curve of time buffer with frame dropping function enabled has a smaller fluctuation and better performance. We can also observe from both Figure 8 and 12 that the server requires some time for the time buffer to settle down in the beginning due to the overestimated default quality levels.

After the time buffer stabilized, the fluctuation was limited within 100 milliseconds, about the interval of three frames, which is tolerable and could be handled by the encoding buffer. After determining the quality level for each encoder, we can re-allocate the bandwidth as shown in Figure 9 and 13. High priority encoders acquire more bandwidth than default to compensate for the drop of quality level; meanwhile, the bandwidth of each low priority encoder is suppressed to meet the total bandwidth limitation. Because there are four low priority encoders to share the retrenchment of bandwidth, the bit rate decrease in them is less than the increase in high priority encoders.

We next compare the PSNR values of different priority encoded sequences in Figure 9 and 13. There are four PSNR curves in each figure. In these figures, "High" represents the result of the high priority encoder, and "Low" is the result of the low priority encoder. "High without bit rate compensation" and "Low without bit rate compensation" are the results of the same quality level sequences in Figure 9 and 13 without bit rate adjustment. The PSNR gap between "High" and "Low" is obvious. The contradiction that the PSNR curve of "Low" is sometimes higher than that of "Low without bit rate compensation" is due to the inaccurate rate control of XviD encoder. Thus, we maintain some extra bandwidth to buffer the excess encoding bit rate. We demonstrate in Figure 15 and 16 the 249th frame of the sequence Football from a high priority encoder and the 249th frame from a low priority encoder.

In the second experiment, we define seven target encoding time to check the frame rate accuracy of our scheme. We use Football as the test sequence, and the result is shown in Table 1. The lower bound of encoding time depends on the complexity of the sequence. Without frame dropping, the lowest encoding time of the sequence Mother and Daughter is 27.07 milliseconds (target 27 milliseconds), and that of Football is 32.98 milliseconds (target 33 milliseconds); with frame dropping, the lowest encoding time of Mother and Daughter decreases to 24.91 (target 25 milliseconds), and that of Football decreases to 29.01 (target 29 milliseconds). The streaming server reaches the lower bound while all encoders have quality level 0, and quality level -1 appears as often as allowed. This means that the server is overburdened, and we can solve the problem by reducing either the number of encoders or loosing the time constraint.

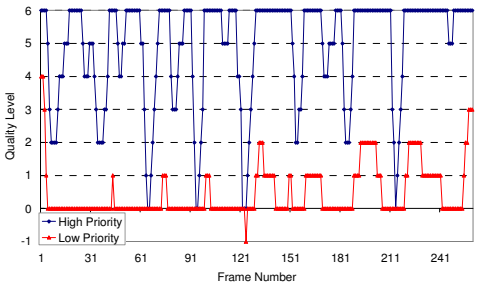

Figure 7. Quality levels for target encoding time 45 ms.

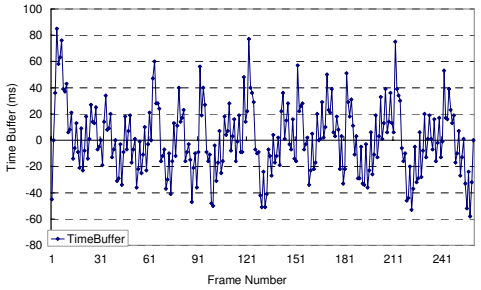

Figure 8. Time buffer for target encoding time 45 ms.

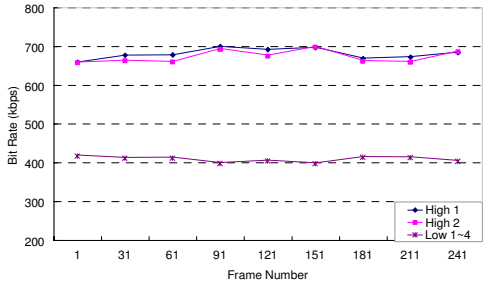

Figure 9. Encoding bit rate for target encoding time 45 ms.

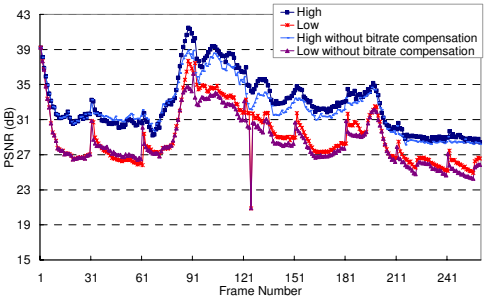

Figure 10. PSNR values for target encoding time 45 ms.

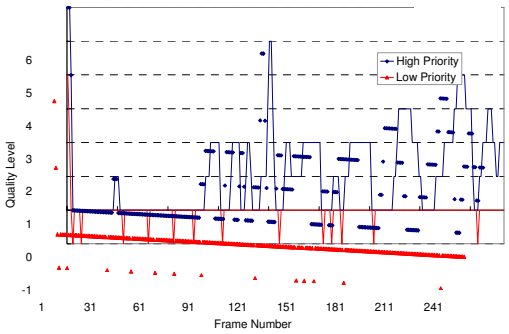

Figure 11. Quality levels for target encoding time 33 ms.

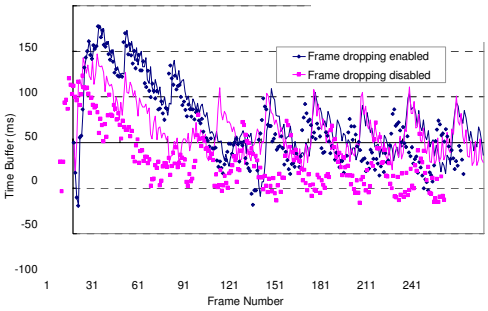

Figure 12. Time buffer for target encoding time 33 ms.

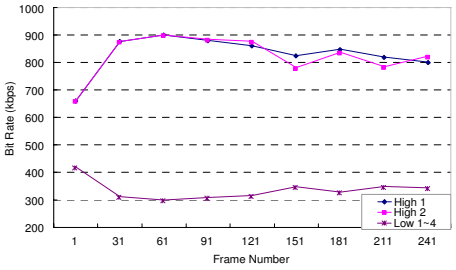

Figure 13. Encoding bit rate for target encoding time 33 ms.

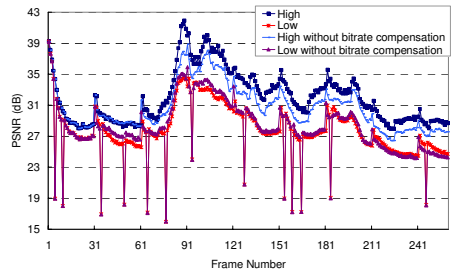

Figure 14. PSNR values for target encoding time 33 ms.

Figure 15. The 249th frame of a high priority encoder.

Figure 16. The 249th frame of a low priority encoder.

Table 1. Average encoding time of Football.

Target Encoding Time	Average Encoding Time without Frame Drop	Average Encoding Time with Frame Drop	Target Frame Rate	Average Frame Rate without Frame Drop	Average Frame Rate With Frame Drop
29 ms	30.44 ms	29.01 ms	34.48 fps	32.89 fps	34.47 fps
31 ms	30.80 ms	30.79 ms	32.26 fps	32.47 fps	32.48 fps
33 ms	32.98 ms	33.08 ms	30.30 fps	30.32 fps	30.23 fps
35 ms	34.77 ms	34.80 ms	28.57 fps	28.76 fps	28.74 fps
40 ms	39.90 ms	39.99 ms	25.00 fps	25.07 fps	25.01 fps
45 ms	44.97 ms	45.14 ms	22.22 fps	22.24 fps	22.16 fps
50 ms	49.98 ms	49.91 ms	20.00 fps	20.01 fps	20.04 fps

5. Conclusion

In this paper, we have presented a complexity-aware live streaming system with bit-rate adjustment for differentiated services. The proposed system is able to allocate the available bandwidth and computational resources to each channel efficiently based on their priorities. The final simulation results show that the system will maintain the timing constraint accurately without the quality drop of high priority channels. This system will be applicable to multimedia home gateways, surveillance systems, IP-based TV, and on-line sports game relay.

6. Acknowledgement

This work was supported in part by the National Science Council of Taiwan under contract NSC 94-2219-E-002-016, and NSC 94-2725-E-002-006-PAE. We would also like to thank Ming-Yu Chen for his help on the implementation of the bit rate adjustment algorithm.

7. References

[1] "Coding of Audio-Visual Objects: Visual", ISO/IEC Standard 14496-2, 1998.

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," *RFC 3550*, IETF Network Working Group, July 2003.

[3] H. Schulzrine, A. Rao, and R.Lanphier, "Real Time Streaming *Protocol* (RTSP)," *RFC 2326*, IETF Network Working Group, April 1998.

[4] M. Handley and V. Jacobson, "SDP: Session Description Protocol," RFC *2327*, IETF Network Working Group, April 1998.

[5] P.-L. Tai, S.-Y Huang, C.-T. Liu, and J.-S. Wang, "Computation-aware scheme for software-based block motion estimation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 9, pp. 901-913, September 2003.

[6] Y.-W Huang, C.-L Lee, C.-Y Chen, and L.-G. Chen, "One-Pass Computation-Aware Motion Estimation with Adaptive Search Strategy," in *Proc. IEEE International Symposium on Circuits and Systems*, Kobe, May 2005.

[7] M.-T. Lu, C.-K Lin, J. Yao, and H. Chen, "Complexity-Aware Live Streaming System," in *Proc. IEEE International Conference on Image Processing*, Genova, September 2005.