JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 24, 1257-1271 (2008)

DOI:10.1688/JISE.2008.24.4.17

Short Paper

The Application-Level Protocol for QoS Supporting in Synchronized Multimedia Sessions

CHUN-CHUAN YANG, SZE-HORNG LEE AND LI-YUAN CHENG
Department of Computer Science and Information Engineering
National Chi Nan University
Puli, 545 Taiwan

An application-level protocol that integrates the concept of application level framing and the network-aware applications is proposed in this paper to support the network applications with the synchronized multimedia session. The application model and the application QoS for the protocol are also proposed. The application QoS for each flow includes the maximum allowable ratio for the lost medium unit and the delay bound of the medium unit associated with the maximum allowable late ratio. The control mechanisms that support the application protocol include error control, real-time control for individual flow within the session, and synchronization control for the multimedia session. Moreover, the application protocol with the control mechanisms provides a quantitative expression for the quality of the synchronized session in terms of the QoS parameters. Measurement results show the effectiveness of the protocol.

Keywords: application-level protocol, synchronized multimedia session, QoS supporting, application level framing, network-awareness

1. INTRODUCTION

Multimedia network applications often require an end-to-end provision of quality of service (QoS) [1, 2]. The requirement results in intensive and vast research for QoS-related issues [3-5]. According to [6], QoS is distinguished to four layers: *user QoS* [7], *application QoS*, *system QoS*, and *network QoS* [8]. The user QoS parameters describe requirements for the perception of multimedia data at the user interface. The application QoS parameters describe requirements for the application services possibly specified in terms of media quality (like end-to-end delay) and media relations (like inter/intrastream synchronization). The system QoS parameters describe requirements on the communications services resulting from the application QoS. These may be specified in terms of both quantitative (like bits per second or task processing time) and qualitative (like multicast, interstream synchronization, error recovery, or ordered delivery of data) criteria. The network QoS parameters describe requirements on network services (like network load or network performance). In this paper, we focus on the application QoS and propose application-level control mechanisms for QoS support in a synchronized multimedia session that does not based on any assumption of the underlying network service.

Received October 14, 2006; revised April 16 & August 23, 2007; accepted December 5, 2007.
Communicated by Homer H. Chen.

Traditional solutions for supporting quality of service for multimedia applications over Internet relied on the functionality of network layer, in which two QoS support framework were proposed: *Integrated Services* (*IntServ*) and *Differentiated Services* (*DiffServ*). The intensive and vast researches for QoS-related issues include the design of the new programming APIs for QoS supporting [20], the resource reservation mechanism [21-23], the QoS management architecture [12, 24, 25], QoS supporting for the wireless environment [10, 11, 14] and telephony QoS issues [26-28], *etc.* In addition to resource management in the network layer, real-time issues were addressed in *Real-time Transport Protocol* (*RTP*) [19] and *RTP Control Protocol* (*RTCP*) by introduction of timing reconstruction at the receiver site. However, the issue of multiple related flows in a session is not addressed in the frameworks/works as mentioned.

Since the network behavior is dynamic and the static allocation strategy (*e.g.*, peak rate assignment for bandwidth requirement) could not achieve the efficiency of the packet switching network (*i.e.*, the gain of the statistical multiplexing), the adaptive network QoS for multimedia transmissions was suggested. Moreover, the best-effort nature or lack of the resource (bandwidth) reservation mechanism for some sub-networks on the transmission path makes it impractical to provide the static end-to-end QoS. However, adaptive QoS implies that once the network situation changes, the application will be notified that the QoS supported by the network subsystem is changed, and it is the responsibility of the application to adapt itself to the new network condition. In other words, adaptive network QoS also requires the adaptability of the application [9-11].

The concept of the middleware mechanism was proposed to reduce the overhead of introducing QoS to the existing multimedia applications such as web servers [9, 12]. For the development of new multimedia network applications, the programmer needs to consider the network dynamics in designing adaptive network applications [13, 14]. Bandwidth measurement mechanisms [15-17] for providing the run-time information about the network condition could be included in the application itself. Such kind of applications was called *network-aware* applications [13].

On the other hand, the concept of *Application Level Framing* (*ALF*) [18, 19] was proposed to integrate the transport protocol functionality in the application. The ALF concept requires that the application controls the packet size in the network and is responsible for the control mechanisms such as error control and real-time control. That is, the ALF concept empowers the application with more controls over network transmissions. Experiments have proved that the ALF concept improves the communications systems performance and allows more advanced techniques for the efficient implementation of communications systems.

Certainly an ALF application should also be a network-aware application since the ALF application controlling network-related functions should also adapt to the network environment. The integration of the concept of ALF and network-awareness is proposed in this paper to support the network applications with the synchronized multimedia session. The QoS parameters for a synchronized session are defined and a couple of application-level control mechanisms including packetization control, error control, real-time control, and synchronization control are proposed in the paper. Performance measurement from a real implementation has demonstrated that the proposed application framework can improve session quality.

The remainder of the paper is organized as follows. The application model and the

session QoS from the application point of view are presented in section 2. The format of the proposed application-level protocol is explained in section 3, and the control mechanisms are presented in section 4. Performance evaluation for the protocol is presented in section 5. Section 6 concludes the paper.

2. APPLICATION MODEL AND SESSION QOS

There are multiple flows within a synchronized multimedia session for a network application. Each flow is responsible for the transmission of one medium. The architecture of the proposed application model for both the sender site and the receiver site of a multimedia session is illustrated in Fig. 1. The data production block in the figure denotes the source of each medium data in the application program at the sender site and is responsible for the input process and the encoding process for one medium. The data unit generated from the data production block is called the medium unit (MU), which could be a video frame or an audio segment, *etc.* MU represents the basic data unit for the encoder/decoder of each medium.

As mentioned in section 1, the middleware under the ALF concept controls the transport protocol functionality such as error control. Since the size of MU is usually too large to be the basic data unit of the control mechanism, we define the basic data unit for the control mechanism as the *application data unit* (*ADU*). Therefore, the middleware segments the MU from the data production block into several fragments for the flow of the medium. Each fragment is further encapsulated by the application protocol format that is presented in the next section. Note that when deciding the size of ADU, we should consider the maximum packet size supported by the underlying network sub-system to eliminate redundant segmentation/reassembly. On the other hand, the size of the ADU cannot be too small to prevent larger header overhead. Normally, we can send probe packets to find out the MTUs of the subnets along the transmission path or simply set the minimum MTU by checking the known MTU information.

The control module at the sender site is responsible for segmenting MUs and adding proper fields of the proposed application protocol in ADUs. In order to provide effective error control, mechanisms of packet-level forward error correction (packet-level FEC) and retransmission under the control of *Application QoS Coordinator* are included in the

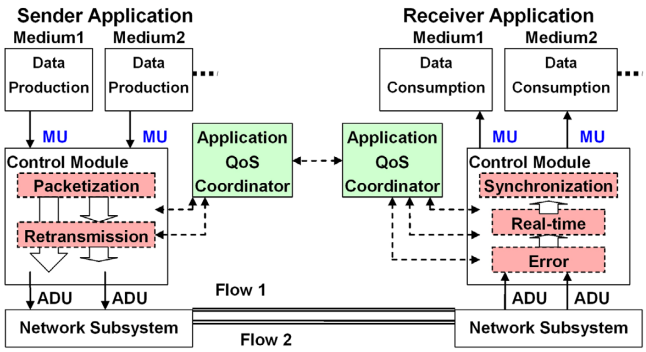

Fig. 1. Application structure for the synchronized multimedia session.

proposed model. On the other side, the control module at the receiver site accepts ADUs from the network subsystem and performs the control mechanisms as well as the reassembling process. As illustrated in Fig. 1, each arrived ADU must pass through error control, real-time control, and synchronization control. Finally, the MUs of all flows in the session are delivered synchronously to the data consumption module of each medium. The Application QoS Coordinator monitors each of the control mechanisms for the sake of session quality and triggers proper actions to cope with performance degradation. Details of these control mechanisms are presented in section 4.

Three parameters ($TxRate^i$, D, $TH\_SuccessGroups$) are specified for QoS requirement of a synchronized session. $TxRate^i$ denotes the transmission rate of the medium data by the sender for flow i that only serves as the input parameter in the flow setup phase for underlying QoS-supported network to reserve proper resource. D denotes the end-to-end delay bound for synchronized groups. $TH\_SuccessGroups$ denotes the ratio requirement of successful synchronized groups. A successful group is defined to be the group that all MUs in the group are delivered to the receiver application within the end-to-end delay bound D. Therefore, $TH\_SuccessGroups$ represents the quantitative requirement over the quality of the multimedia session.

In order to support proposed control mechanisms, three system parameters are defined and monitored for a multimedia session: $SuccessGroups_{(K)}$, $SuccessMUs^i_{(K)}$, and $EstRTT^i$. $SuccessGroups_{(K)}$ is used to record the run-time ratio of the successful groups in the last K groups, $SuccessMUs^i_{(K)}$ is the run-time ratio of MUs (in the last K MUs) for *flow i* that are successfully reassembled from ADUs within the end-to-end delay bound D, and $EstRTT^i$ is the estimated round-trip time for flow i. Two sets of the system parameters are used: the case of ($K = \infty$) and the case of (K = a finite integer, *e.g.* $K = 100$). The value of a system parameter with ($K = \infty$) represents the aggregate (average) behavior of the parameter starting from the beginning of the session, while the other case represents the transient behavior by observing only a short period of time in the past. As will be explained in section 4, by comparing the values of these system parameters under different K values, the reason for performance degradation can be identified to trigger effective control actions. The summary of the QoS parameters as well as system parameters is displayed in Fig. 2.

3. PROTOCOL FORMAT

As mentioned in section 2, each MU is segmented to several fragments, which are further encapsulated into ADUs. The header of each ADU is responsible for carrying information for the control mechanisms at the receiver site. The format of the ADU is depicted in Fig. 3. The field of *FlowID* is used for the application to distinguish the flows in the session. The *Sequence#* field carries information for reassembling the original MU. There are three sub-fields in the *Sequence#* field: *MU#*, *Fragment#*, and *NumberOfFragments*. The field of MU# represents the ID of the medium unit that is the source of this ADU. The field of *Fragment#* indicates the position of this ADU in the original MU. The *NumberOfFragments* field indicates how many ADUs belong to the same MU. That is, all the ADUs, which are from the same MU, will have the same value of *MU#* and *NumberOfFragments*. The #GroupMembers field shows the number of MUs in a synchronous group.

QoS Parameters:
$TxRate^i$: Transmission rate of the medium data for flow *i*.
D : End-to-end delay bound of MU for a synchronized session. (For all flows in the same synchronized session have the same *D*)
TH_SuccessGroups: The threshold of minimum synchronized session.
System Parameters:
$SuccessGroups_{(K)}$: The run-time ratio of the successful groups in the last *K* groups. Two sets of value of K are used that K= ? and K=a finite integer (e.g. K=100)
$SuccessMUs^i{}_{(K)}$: The run-time ratio of MUs in the last *K* MUs for flow *i*. Two sets of value of K are used that K= ? and K=a finite integer (e.g. K=100)
$EstRTT^i$: The estimated round-trip time for flow *i*.

Fig. 2. The QoS parameters and system parameters in a synchronized multimedia session.

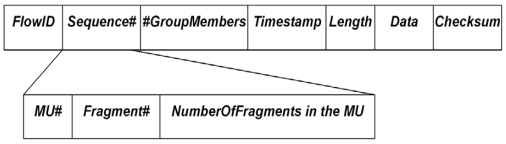

Fig. 3. The application protocol format.

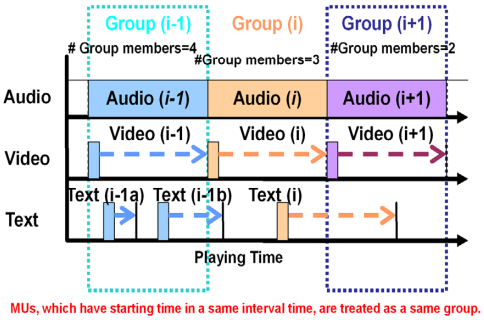

Fig. 4. The synchronized group members.

The field of *Timestamp* carries the generation time of the original MU. That is, the ADUs from the same MU have the same value of *Timestamp*. The *Timestamp* field is used for the real-time control mechanism and the synchronization control mechanism at the receiver site. As will be explained in next section, the MUs with the same *Timestamp* from different flows must be synchronized in the synchronization control mechanism. The *Length* field indicates the length of the following *Data* field. The *Data* field carries the fragment data segmented from the MU. Finally, the *Checksum* field allows the flow with tighter error checking function. The sender and receiver must decide if the *Checksum* field is needed or not in the flow setup phase.

Fig. 4 shows that how a synchronous group is formed. Audio MU ($i - 1$), video MU ($i - 1$), text MU ($i - 1a$) and text MU ($i - 1b$), which has the starting time in a same interval time respectively, are grouping into the same synchronous group, group ($i - 1$). Group ($i - 1$) has four group members, group (i) has three group members and group ($i + 1$) has only two group members. We pre-defined a very small interval time for grouping MUs, but for simplicity we can also use audio MU as grouping reference as in our implementation.

4. CONTROL MECHANISMS

In addition to the encapsulation of ADUs, there are two mechanisms proposed in the control module at the sender site to cope with the degradation of session quality as mentioned in section 2: *packet-level FEC* and *retransmission*. The Application QoS Coordinator determines when and which mechanisms to be performed by investigating the values of QoS parameters (*i.e. TH_SuccessGroups* and *D*) as well as system parameters (*i.e.* $SuccessGroups_{(K)}$, $SuccessMUs^i_{(K)}$ and $EstRTT^i$) presented in section 2. In the following, we focus on the control mechanisms and explain the algorithm adopted in the proposed Application QoS Coordinator.

4.1 Error Control

The objective of the error control is to detect if there is some lost ADUs for each flow by checking the in-sequence. Therefore, the main actions of the error control include (1) checking the *Sequence#* field for the incoming ADUs, (2) checking the data integrity if the optional checksum is enabled for the flow, (3) passing the information of the detected lost or damaged ADUs to the Application QoS Coordinator. The process of the error control is illustrated in Fig. 5.

There are two kinds of errors for the ADU, checksum error and loss. The checksum error is detected via the *OptionalChecksum* field, and the lost ADUs are detected by the *in-sequence* checking. In order to make the in-sequence checking work correctly for the detection of lost ADUs, we assume that all the ADUs of the same flow follow the same path from the sender to the receiver, which is actually the concept of *virtual circuit*. The assumption is reasonable for the multimedia session with long duration.

4.2 Real-time Control

The real-time control process reassembles the ADUs to the original MU and checks if the end-to-end delay of the MU is smaller than the end-to-end delay bound. We define one MU as the *late MU* if the real-time control process could not finish reassembling the MU within the delay bound. That is, all ADUs (including the retransmission of the lost ADU) form the same MU must arrive to the receiver within the delay bound. In order to compute the end-to-end delay of MUs correctly at the receiver site, the sender and the receiver must use a common global time reference. The receiver uses the *Timestamp* value of ADUs and the delay bound to decide the local expiration time (*i.e. Timestamp + D*) for the reassembly of the MU as illustrated in Fig. 6. If the real-time control process could not finish the reassembly of one MU before the expiration time, all ADUs of the MU are dropped, and the values of $SuccessMUs^{i}_{(\infty)}$ and $SuccessMUs^{i}_{(K)}$ are updated.

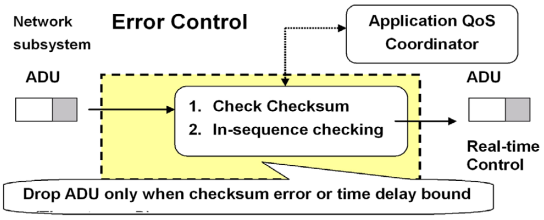

Fig. 5. The error control mechanism.

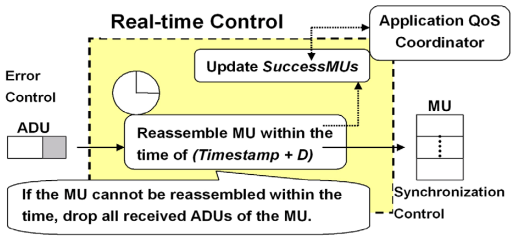

Fig. 6. The real-time control mechanism.

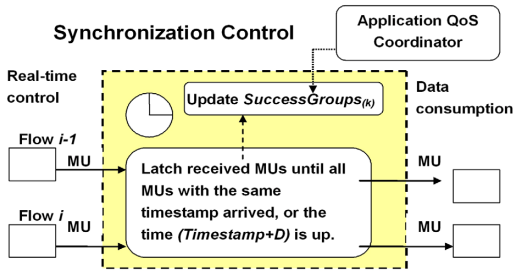

Fig. 7. The synchronization control mechanism.

4.3 Synchronization Control

The only function of synchronization control is to synchronize the delivery of all MUs with the *Timestamp* within a small and same interval time (synchronized groping time). Therefore, the process latches the MUs with the same *Timestamp* until the MUs of all flows arrive or the local time (*Timestamp* + *D*) is up. In other words, all MUs with the same interval time must be synchronously delivered to the data consumption module within the delay bound (*D*). Moreover, the values of $SuccessGroups_{(K)}$ and $SuccessGroups_{(\infty)}$ are updated in this stage. The process for the synchronization control is illustrated in Fig. 7.

4.4 Application QoS Coordinators

Application QoS Coordinator is responsible for monitoring the quality of the session and triggering proper action to cope with quality degradation. Application QoS Coordinator must determine when and which flow to trigger proper mechanism (packet-level FEC and retransmission) for quality improvement. The following rules are adopted:

(1) If ($SuccessGroups_{(\infty)} \geq TH\_SuccessGroups$), no actions are triggered.
(2) If ($SuccessGroups_{(\infty)} < TH\_SuccessGroups$), it implies that the long-term quality value of the session does not satisfy the user's requirement, thus proper actions should be made.

Application QoS Coordinator first identifies the flows of performance bottleneck and triggers the proper mechanism. Following rules are applied in this case:

(A) The flow such that $SuccessMUs^{i}_{(K)} < SuccessMUs^{i}_{(\infty)}$ is identified as the flow of performance bottleneck, since the quality of the flow is getting worse.
(B) For each flow of performance bottleneck, if $2 * EstRTT^{i} \leq D$ implying that retransmission of the lost ADUs is feasible to improve the ratio of successful MUs, the mechanism of retransmission for the lost ADUs is requested by Application QoS Coordinator. On the other hand, if $2 * EstRTT^{i} > D$, packet-level FEC is triggered instead.

If Application QoS Coordinator has already triggered actions for performance improvement for a period of time and the new value of $SuccessGroups_{(K)}$ is still less than $SuccessGroups_{(\infty)}$ implying that the mechanisms did not work well. In this case, Application QoS Coordinator will terminate the multimedia session (in the case of best-effort underlying network subsystem) or re-negotiate with the underlying QoS-supported network subsystem for better resource reservation.

4.5 Slow-start and Slow-stop

More fluctuations of the network situation often occur in the beginning of a multimedia session [29, 30]. In order to avoid frequently entering the re-negotiation state and termination of the session, Application QoS Coordinator should delay the execution of the control rules and get rid of unnecessary investigation of the system parameters in the

beginning of each flow. The strategy is called *slow-start*. The slow-start strategy is reasonable since the user should be able to tolerate more quality degradation in the beginning of the session. On the other hand, after the session reaches the steady state, if a sudden congestion happens and the QoS of the session is violated. The session should delay the re-negotiation (termination) process since the user would tolerate the QoS degradation resulted from the transient congestion. The strategy is called slow-stop. The time for postponement of the re-negotiation (termination) process depends on the characteristics of the application and the medium type of the flow.

5. PERFORMANCE MEASUREMENT

5.1 Implementation and Performance Criteria

We have implemented the proposed protocol as well as the associated mechanisms by using Microsoft Visual C++ 6.0 on Windows 2000. The medium used in the synchronized session includes Motion-JPEG video, PCM audio, and whiteboard data. Two computers (Pentium III 800M Hz with 192MB memory and Pentium III 1GHz with 256MB memory) each equipped with a web cam, a sound card and a speaker act as the sender and the receiver. The resolution of the video frame is 160 × 120 (QQVGA format). Generation rates for audio and video are both 10 MU/sec, the rate for HTML is 1 MU/2sec. Moreover, one audio MU is encapsulated into one ADU, and so is whiteboard data. One video MU is encapsulated into a couple of ADUs that ranges in between 2 ~ 6 ADUs.

In order to simulate the behavior of a WAN on a local area network, we added a gateway process in the receiver computer to perform proper operations on every packet it receives, which results in a similar behavior as if the packets were transmitted across a WAN. The configuration of the WAN emulator is illustrated in Fig. 8. A probability model as well as a normally distributed delay model $N(\mu, \sigma)$ are implemented by the gateway process. Each packet received by the gateway process is to be dropped with a probability (0.05, modeling packets are dropped due to the physical link error), or be delayed for some time before forwarding the packet to the receiving process. The delay time for each packet is generated by the delay model.

Three criteria are used for performance evaluation: (1) *the jitters of the synchronous group*, (2) the *time offset* between the sender and the receiver, and (3) *the ratio of failed groups*. The jitters of a synchronous group are defined as the maximum offset of the delivery time of MUs in the same group. The time offset is defined as the time difference between the generation time of an MU at the sender site and the playback time of the MU at the receiver site. Thus, the time offset mainly consists of the packetization delay, transmission delay in the network, and the buffering time at the receiver. A large value of time offset between the sender and the receiver does not fit in interactive applications that demand less real-time communications. On the other hand, a small value of time offset can easily result in empty buffer and re-buffering of MUs for smooth playback is necessary. Therefore, it is better to keep the value of time offset as stable as possible. Lastly, the ratio of failed groups represents the quality of the session.

In order to evaluate the performance of the proposed protocol, contrasts without the support of the proposed mechanism were also implemented. As illustrated in Fig. 9, there are two cases of contrast for performance comparison: *TCP-based contrast* and *UDP-*

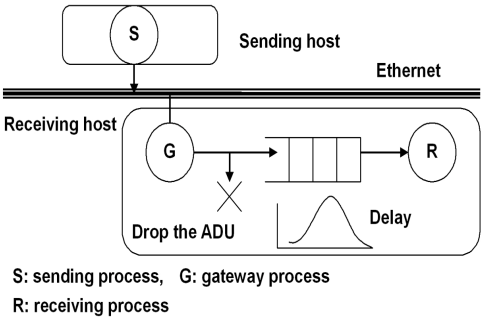

Fig. 8. WAN emulator.

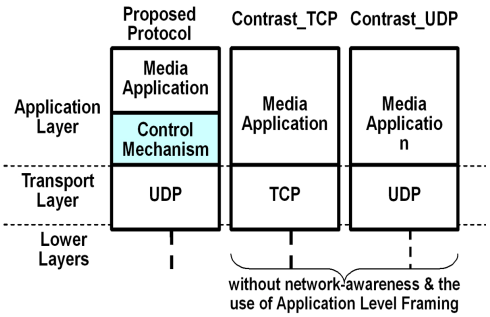

Fig. 9. The proposed protocol and the two contrast protocols.

based contrast. Contrast_UDP presents the case that multimedia applications directly base on UDP in which there is no retransmission mechanism for error control. On the other hand, the retransmission mechanism for reliable transmission in Contrast_TCP is mandatory. Moreover, there is no any application level control mechanism in both contrasts. Moreover, for the sake of simply investigating the performance of the proposed mechanisms, the QoS parameters were set as ($TH\_SuccessGroups$ = 100%, D = 1000ms), and no termination of the session during the measurement process.

5.2 Jitters of a Synchronous Group

Figs. 10-11 and Figs. 12-13 display the jitters of the synchronous group for the proposed protocol and two contrasts under different network conditions, respectively. The proposed protocol shapes the arrival pattern into a synchronous pattern (*i.e.* jitters = 0). Out-of-real-time MUs are discarded. On the other hand, the two contrasts without the support of synchronization control result in the phenomenon of out-of-synchronization, and as the network condition gets worse (*e.g.* standard deviation changes from 50ms to 200ms in Figs. 10-11), the phenomenon of out-of-synchronization is getting serious.

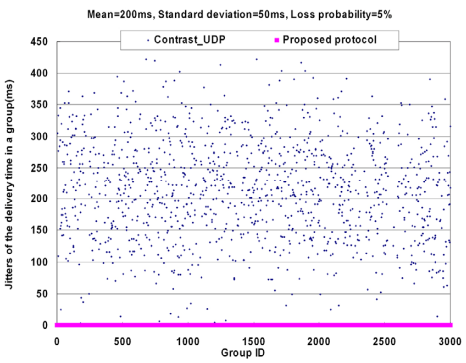

Fig. 10. The jitters of a synchronous group for proposed protocol and Contrast_UDP (standard deviation = 50ms).

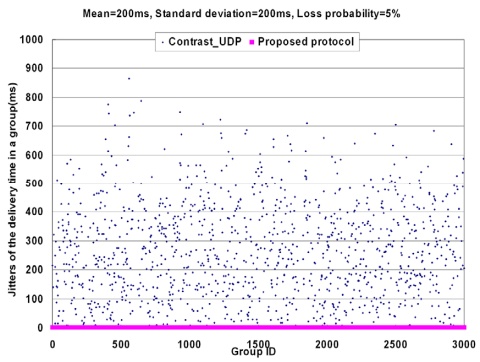

Fig. 11. The jitters of a synchronous group for proposed protocol and Contrast_UDP (standard eviation = 200ms).

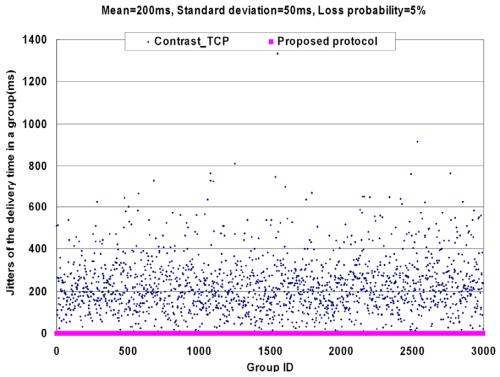

Fig. 12. The jitters of a synchronous group for proposed protocol and Contrast_TCP (standard deviation = 50ms).

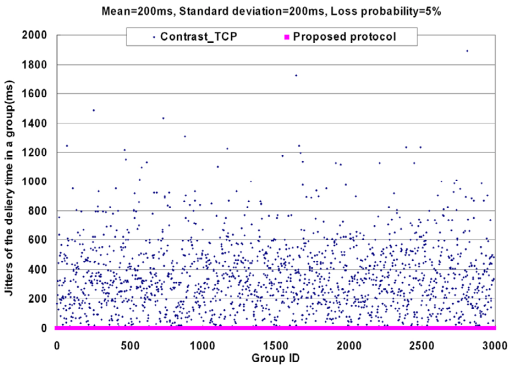

Fig. 13. The jitters of a synchronous group for proposed protocol and Contrast_TCP (standard deviation = 200ms).

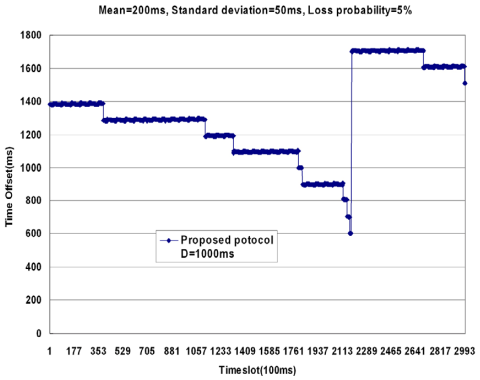

Fig. 14. The time offset (TO) changes for proposed protocol with standard deviation = 50ms.

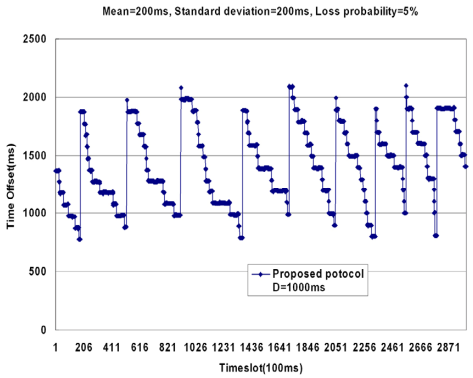

Fig. 15. The time offset (TO) changes for proposed protocol with standard deviation = 200ms.

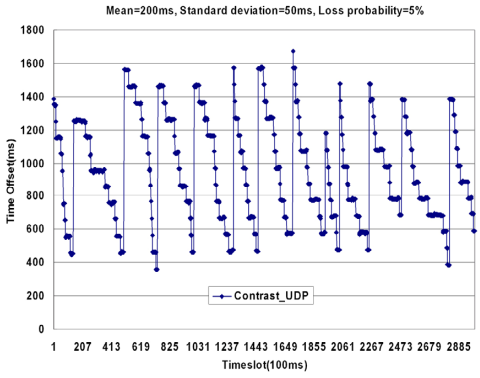

Fig. 16. The time offset (TO) changes for Con trast_ UDP with standard deviation = 50ms.

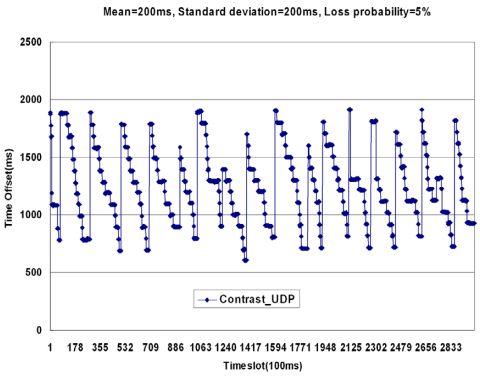

Fig. 17. The time offset (TO) changes for Contrast_ UDP with standard deviation = 20ms.

5.3 Time Offset

Figs. 14-15 display the time offset of the proposed protocol under different network conditions. Dropping on the curves of the time offset is due to skipping the playback of unsuccessful groups in the case that some MUs in the groups are lost or out-of-real-time. As the time offset drops under a predefined threshold value, re- buffering is triggered by the player for continuation of playback and the value of time offset jumps to a much higher level. Therefore, a slower dropping pattern of time offset and fewer number of re-buffering imply a better performance. By comparing Figs. 14-15 with Figs. 16-17, the performance of Contrast_UDP in terms of time offset is worse than the proposed protocol because of the larger number of re-buffering as well as the faster dropping pattern of time offset. The reason behind the better performance of the proposed protocol over Contrast_UDP is due to the remedy of lost ADUs by proposed error control mechanisms (on-demand retransmission and packet-level FEC).

As displayed in Figs. 18-19, the time offset of Contrast_TCP never goes down during the session because of reliable transmission in TCP. Since lost packets are always retransmitted, all MUs will eventually arrive in the receiver's buffer such that the player has to wait for all MUs in a group before playback. As a consequence, waiting of the retransmitted packets results in the inevitable increase of the time offset.

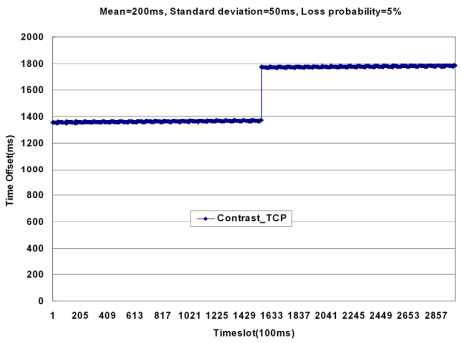

Fig. 18. The time offset (TO) changes for Contrast_ TCP with standard deviation = 50ms.

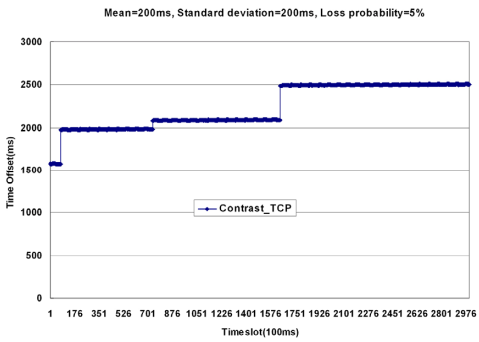

Fig. 19. The time offset (TO) changes for Contrast _ TCP with standard deviation = 200ms.

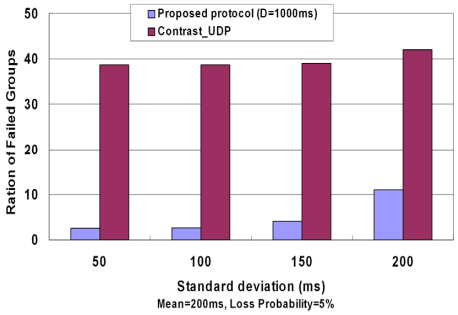

Fig. 20. Performance for Contrast_UDP and our proposed protocol.

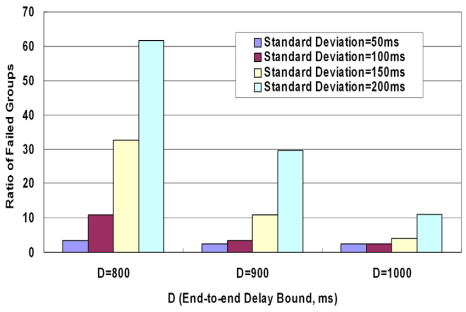

Fig. 21. Performance for different *D* and network delay.

5.4 Ratio of the Failed Group

A synchronous group is defined as a failed group if one or more MUs in the group are lost or out-of-real-time. Fig. 20 displays the ratio of the failed group of the proposed protocol and Contrast_UDP under different network conditions. Because of the enhancement of error control mechanisms, the proposed protocol outperforms Contrast_UDP in terms of smaller ratio of failed groups. As shown in Fig. 21, a smaller end-to-end delay bound results in the increase of the ratio of failed groups since the operation space for the proposed error control mechanisms shrinks.

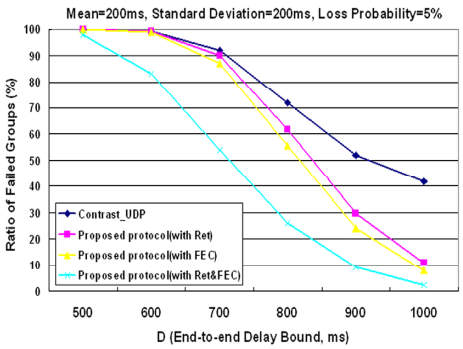

Fig. 22. Ratio of failed groups: retransmission vs. FEC.

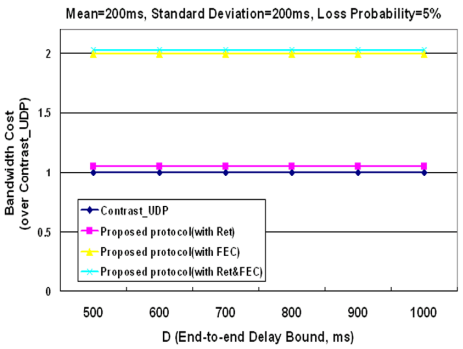

Fig. 23. Bandwidth cost of the error control schemes.

At last, the performance measurements of retransmission or packet-level FEC in our proposed protocol are shown in Fig. 22 under different end-to-end delay bounds. The packet-level FEC in our implementation only transmits the same packet twice for simplicity. By comparing FEC and retransmission, there is better improvement (3 ~ 5%) for FEC when end-to-end delay bound larger than 700ms. It is a reasonable time delay for retransmission operation. The main drawback of FEC is wasting network bandwidth. Great improvement is obtained for cooperation of control mechanisms, FEC and retransmission, but accompanied with the highest bandwidth cost as shown in Fig. 23. Lastly, different FEC encoding schemes have different correction rate and extra bandwidth cost, therefore, under the same correction rate, the less the extra bandwidth required, the better the FEC scheme.

6. CONCLUSION

Multimedia over Internet has become very popular in recent years. QoS support in transmission is necessary to meet real-time demands of multimedia data. Previous work in QoS support over Internet mainly focused on providing QoS mechanisms for a single medium flow in the network layer. In this paper, we integrate the concept of *Application-Level Framing* and *Network-Awareness* to propose an application-level protocol for a synchronized multimedia session with multiple flows. The application model as well as

the QoS parameters for a synchronized session is presented in the paper. Application adaptability to network condition is provided by incorporating adaptive control mechanisms in the proposed protocol including *packetization* and *retransmission* at the sender site, *error control*, *real-time control*, and *synchronization control* at the receiver site, and *Application QoS Coordinator*. Application QoS Coordinator is responsible for monitoring the quantitative quality of the session and triggering proper control mechanisms to cope with quality degradation. Measurements from a real implementation have demonstrated that a better performance can be achieved by the proposed protocol over contrasts in terms of jitters in the synchronous group, time offset between the sender and the receiver, and the ratio of successful groups.

As to this proposed protocol, there is further discussion for servicing multicast. We believe that it may work better on application-layer multicast overlay network rather than IP multicast network. In application-layer multicast, end-hosts participating in a multicast group, organize themselves into overlay spanning trees for data delivery. Proper control mechanisms can be triggered by QoS Coordinator among these end-hosts which act as pairs of sender and receivers.

REFERENCES

1. A. Vogel, *et al.*, "Distributed multimedia and QoS: a survey," *IEEE Multimedia*, Vol. 2, 1995, pp. 10-19.
2. L. Skorin-Kapov, *et al.*, "End-to-end QoS for virtual reality services in UMTS," in *Proceedings of the 7th International Conference on Telecommunications*, 2003, pp. 337-344.
3. C. C. Yang and J. H. Huang, "A multimedia synchronization model and its implementation in transport protocols," *IEEE Journal of Selected Area in Communications*, Vol. 14, 1996, pp. 212-225.
4. X. Li, S. Paul, and M. Ammar, "Layered video multicast with retransmission (LVMR): evaluation of error recovery scheme," in *Proceedings of IEEE 7th International Workshop on Network and Operating System Support for Digital Audio and Video*, 1997, pp. 161-171.
5. S. Tasaka, *et al.*, "Live media synchronization quality of a retransmission-based error recovery scheme," in *Proceedings of IEEE International Conference on Communications*, 2000, pp. 1535-1541.
6. R. Steinmetz and K. Nahrstedt, *Multimedia: Computing*, *Communications and Application*, Englewood Cliffs, Prentice-Hall, New Jersey, 1995.
7. Z. Wang, *et al.*, "Studying streaming video quality: from an application point of view," in *Proceedings of the 11th ACM International Conference on Multimedia*, 2003, pp. 327-330.
8. S. P. Chan, *et al.*, "Multimedia streaming gateway with jitter detection," *IEEE Transactions on Multimedia*, Vol. 7, 2005, pp. 585-592.
9. S. Brandt, *et al.*, "A dynamic quality of service middleware agent for mediating application resource usage," in *Proceedings of 19th IEEE Real-Time Systems Symposium*, 1998, pp. 307-317.
10. G. Lin and C. K. Toh, "Performance evaluation of a mobile QoS adaptation strategy

for wireless ATM networks," in *Proceedings of IEEE International Conference on Communications*, 1999, pp. 744-748.
11. A. Kassler and P. Schulthess, "Extending the QoS paradigm of wireless ATM to mobile broadband applications and to the user," in *Proceedings of IEEE Wireless Communications and Networking Conference*, 1999, pp. 368-372.
12. T. F. Abdelzaher and K. G. Shin, "QoS Provisioning with *q*Contracts in web and multimedia servers," in *Proceedings of the 20th IEEE Real-Time Systems Symposium*, 1999, pp. 44-53.
13. J. Bolliger and T. Gross, "A framework-based approach to the development of network-aware applications," *IEEE Transactions on Software Engineering*, Vol. 24, 1998, pp. 376-390.
14. H. Liu and M. E. Zarki, "Adaptive delay and synchronization control for Wi-Fi based AV conferencing," in *Proceedings of the 1st International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, 2004, pp. 84-91.
15. J. Bolliger, T. Gross, *et al.*, "Bandwidth modelling for network- aware applications," in *Proceedings of IEEE Conference on Computer Communications*, Vol. 3, 1999, pp. 1300-1309.
16. K. Lai and M. Baker, "Measuring bandwidth," in *Proceedings of IEEE Conference on Computer Communications*, Vol. 1, 1999, pp. 235-245.
17. V. Paxson, "End-to-end internet packet dynamics," *IEEE/ACM Transactions on Networking*, Vol. 7, 1999, pp. 277-292.
18. D. Clark and D. Tenenhouse, "Architectural considerations for a new generation of protocols," in *Proceedings of ACM Special Interest Group on Data Communication*, Vol. 20, 1990, pp. 200-208.
19. H. Schulzrinne, *et al.*, "RTP: a transport protocol for real-time applications," RFC 3550, 2003.
20. P. G. S. Florissi, *et al.*, "QoSockets: a new extension to the sockets API for end-to-end application QoS management," in *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management*, 1999, pp. 655-668.
21. R. Braden, *et al.*, "Resource ReSerVation protocol (RSVP)-version 1 functional specification," Internet Draft, Internet Engineering Task Force, 1996.
22. P. Y. Wang, *et al.*, "Experimental QoS performance of multimedia applications," in *Proceedings of the 19th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies*, 2000, pp. 970-979.
23. I. Foster, *et al.*, "A distributed resource management architecture that supports advance reservations and co-allocation," in *Proceedings of the 7th IEEE International Workshop on Quality of Service*, 1999, pp. 27-36.
24. A. G. Malamos, *et al.*, "On the definition, modeling, and implementation of quality of service (QoS) in distributed multimedia systems," in *Proceedings of IEEE International Symposium on Computers and Communications*, 1999, pp. 397-403.
25. S. Chatterjee, *et al.*, "Modeling applications for adaptive QoS-based resource management," in *Proceedings of High-Assurance Systems Engineering Workshop*, 1997, pp. 194-201.
26. G. Araniti, *et al.*, "Multimedia traffic QoS adaptation in UMTS systems through a middleware functionality," in *Proceedings of IEEE International Conference on Communications*, Vol. 1, 2003, pp. 17-21.

27. E. Exposito, *et al.*, "The XQoS aware and fully programmable transport protocol," in *Proceedings of the 11th IEEE International Conference on Network*s, 2003, pp. 249-254.
28. X. Chen, *et al.*, "Survey on QoS management of VoIP," in *Proceedings of International Conference on Computer Networks and Mobile Computing*, 2003, pp. 69-77.
29. C. C. Yang and C. F. Liu, "A bandwidth-based polling scheme for QoS support in bluetooth," *Journal of Computer Communications*, Vol. 27, 2004, pp. 1236-1247.
30. C. C. Yang and K. Y. Lin, "Distributed mobile tracking: a novel location management scheme for routing improvement in cellular IP networks," *Journal of Computer Networks*, Vol. 43, 2003, pp. 141-167.

Chun-Chuan Yang (楊峻權) received his B.S. degree in Computer and Information Science from National Chiao Tung University, Taiwan, in 1990 and Ph.D. degree in Computer Science from National Taiwan University in 1996. He joined the Department of Computer Science and Information Engineering, National Chi Nan University (NCNU), Puli, Taiwan, as an Assistant Professor in 1998. Since August 2003, he has been an Associate Professor. His research area of interests includes multimedia network protocols, multimedia synchronization control, and multimedia applications.

Sze-Horng Lee (李思宏) received his B.S. degree in Information Computer and Education from National Taiwan Normal University, Taiwan, in 1995 and M.S. degree in Information Management from National Chi Nan University in 1999. Currently, he is a Ph.D. candidate of Department of Computer Science and Information Engineering, National Chi Nan University. His research interests are multimedia network protocols, multimedia synchronization control, and network management.

Li-Yuan Cheng (鄭力源) received his B.S. degree in Computer Science and Information Engineering from I-Shou University, Taiwan, in 1997 and M.S. degree in Computer Science and Information Engineering from National Chi Nan University in 2005. His research interest is synchronized multimedia.