Design of the Application-Level Protocol for Synchronized Multimedia Sessions

Chun-Chuan Yang
Multimedia and Communications Laboratory
Department of Computer Science and Information Engineering
National Chi Nan University, Taiwan, R.O.C.
ccyang@csie.ncnu.edu.tw

***Abstract*- 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 connection includes the maximum allowable ratio of 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 connection 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. Simulation results show the effectiveness of the protocol.**

I. INTRODUCTION

Multimedia network applications often require an end-to-end provision of quality of service (QoS) [1]. The requirement results in intensive and vast research for QoS-related issues. The QoS-related issues include the design of the new programming APIs for QoS supporting [2], the resource reservation mechanism [3, 4, 5], the QoS management architecture [6, 7, 8, 9], and QoS supporting for the wireless environment [10, 11] etc. 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 [6, 7, 8, 9, 10, 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 [8, 9]. For the development of new multimedia network applications, the programmer needs to consider the network dynamics in designing adaptive network applications [14]. Bandwidth measurement mechanisms [15, 16, 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 [14].

On the other hand, the concept of *Application Level Framing (ALF)* [12, 13] 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 application QoS is defined for each connection within the multimedia session, and the application-level protocol with the corresponding control mechanisms, which include error control, real-time control, and synchronization control, is proposed.

The remainder of the paper is organized as follows. The application model and the connection QoS from the application point of view are presented in section II. The format of the proposed application-level protocol is explained in section III, and the control mechanisms are presented in section IV. Performance evaluation for the protocol is presented in section V. Section VI concludes the paper.

II. APPLICATION MODEL AND QOS

There are multiple connections for a network application with the synchronized multimedia session. Each connection is responsible for the transmissions 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 I, the application 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 application segments the MU from the data production block into several fragments for the connection 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.

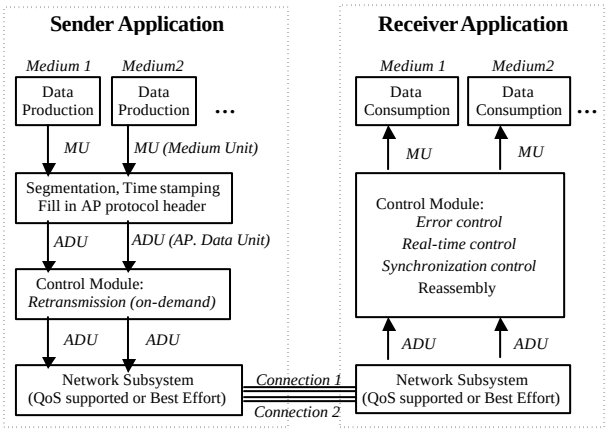


Fig. 1. Application structure for the synchronized multimedia session

The only control mechanism supported by the application at the sender site is the on-demand retransmission for error control, which means the sender only retransmits ADU as requested by the receiver. Since the sender usually acts as the data server for multiple concurrent receivers, the on-demand policy of retransmission reduces the overhead of the sender to support ALF.

The application model does not impose any QoS constraint on the underlying network subsystem, which means the underlying network could be either the QoS supported network or the best-effort network. The difference between the best-effort network and the QoS supported network is the different actions taken by the application when the application QoS is violated. Once the application QoS is violated, the application enters the re-negotiation state of the connection for the QoS supported network, but terminates the connection for the best-effort network. The interface between the application and the QoS supported networks is not addressed in the paper.

On the other hand, the receiver site accepts ADUs from the network subsystem and performs the control mechanisms as well as the reassembling process, which are explained in section IV. As illustrated in Fig. 1, each ADU from the network must pass error control, real-time control, and synchronization control in the receiver application. Finally, the MUs of all connections in the session are delivered synchronously to the data consumption module of each medium.

Five parameters are used to specify the QoS requirement for each connection within the synchronized session. $TxRate^i$ denotes the transmission rate of the medium data by the sender for connection *i*. $MaxP^i_{loss}$ denotes the maximum allowable loss ratio of MU for connection *i*. P^i_{loss} denotes the loss rate that triggers the on-demand retransmission mechanism. Furthermore, the value of P^i_{loss} is required to be smaller than the value of $MaxP^i_{loss}$. D^i denotes the end-to-end delay bound of MU for connection *i*, and P^i_{late} denotes the maximum allowable ratio of *late MUs*. The MU with the end-to-end delay exceeding D^i is defined as the *late MU*. The summary of the application QoS parameters for a connection is shown in Fig. 2. Notice that each connection in the same session should have the same value of the end-to-end delay bound because of the the synchronization requirement, i.e. for all connection *i* in the synchronized session, $D^i = D$.

All of the parameters except $TxRate^i$ are used for the control mechanisms of the proposed application-level protocol. $TxRate^i$ only serves as the input parameter in the connection setup phase

$TxRate^i$: *Transmission rate of the medium data for connection i.*

P^i_{loss} : *Loss rate that triggers retransmission mechanism for connection i.*

$MaxP^i_{loss}$: *Maximum loss rate of medium unit (MU) allowed for connection i.*

P^i_{late} : *Maximum rate of late MU allowed for connection i.*

D^i : *End-to-end delay bound of MU for connection i.*

For all connection i in the synchronized session, $D^i = D$

Fig. 2. Application QoS for synchronized multimedia sessions

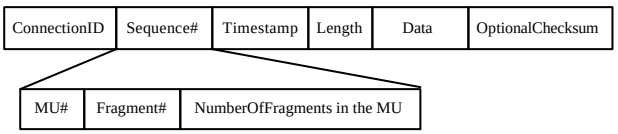


Fig. 3. The application protocol format

for underlying QoS-supported networks to reserve proper resource. More specifically, ($TxRate^i$, $MaxP^i_{loss}$, D^i, P^i_{late}) could be passed to the underlying network system in the setup phase of each connection if the underlying network is QoS-supported.

III. PROTOCOL FORMAT

As mentioned in section II 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 *ConnectionID* is used for the application to distinguish the connections 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 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 connections 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 *OptionalChecksum* field allows the connection with tighter error checking function. The sender and receiver must decide if the *OptionalChecksum* field is needed or not in the connection setup phase.

IV. CONTROL MECHANISMS

In this section, three control mechanisms preformed at the receiver site are presented. The execution sequence for the mechanisms is, first the error control, second the real-time control, and finally the synchronization control. Reassembly of MUs is executed during the process of the real-time control. Two application variables are defined for these mechanisms, R^i_{loss} and R^i_{late}. R^i_{loss} is used to record the run-time loss ratio of MUs, and R^i_{late} is used to record

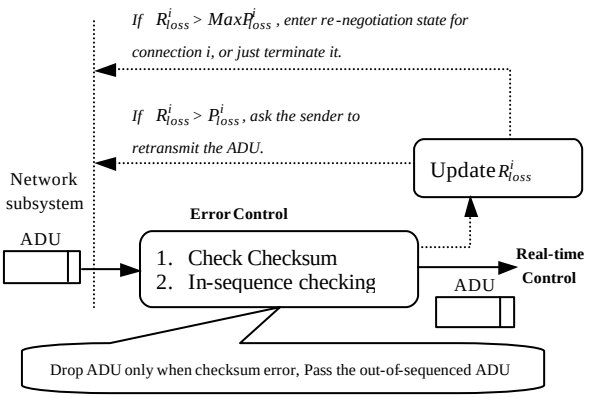


Fig. 4. The error control mechanism.

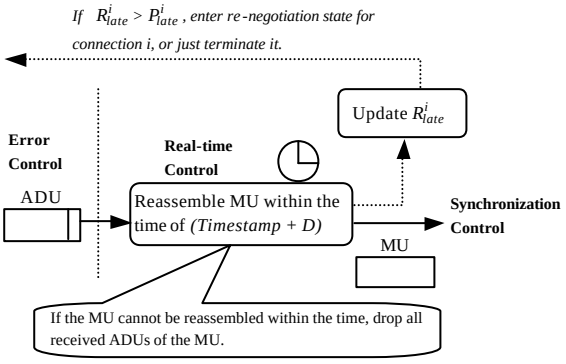


Fig. 5. The real-time control mechanism.

the run-time late ratio of MUs. The control mechanisms uses the two variables to examine whether the application QoS is voilated or not and to take the proper action for the violation. We explain each of the control mechanisms in the following.

4.1 Error control

The objective of the error control is to detect if there is some lost MU, which results from the lost ADU of the MU, and to compute the new value for R^i_{loss}. Therefore, the main functions of the error control include checking the *Sequence#* field for the incoming ADUs, checking the data integrity if the optional checksum is required for the connection, calculating the new value for R^i_{loss}, and finally triggering the proper action according the new value of R^i_{loss}. The process of the error control is illustrated in Fig. 4.

Since the calculation of R^i_{loss} is based on MUs, the error control process has to decide if some fragments of the MU are lost or in error according to the incoming ADUs. There are two kinds of errors for the MU, checksum error of ADUs and loss of ADUs. 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 connection 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 connection with long duration.

Therefore, by examining the *Sequence#* field of the incoming ADUs, the error control process could detect the lost MUs, and calculate the value of R^i_{loss}, which is the ratio of the number of lost MUs over the total number of MUs that should be received. The new value of R^i_{loss} is then compared with the values of $MaxP^i_{loss}$ and P^i_{loss}. If $R^i_{loss} > MaxP^i_{loss}$, which means the QoS is voilated, the connection should enter the re-negotiation state or be terminated. If $P^i_{loss} < R^i_{loss} \le MaxP^i_{loss}$, it triggers the retransmission mechanism for the lost MU. The application issues a request for retransmission of the lost MU, and re-compute R^i_{loss} by assuming the retransmitted MU will arrive correctly. Note that the error control process does not drop any ADU with the correct checksum. Thus, the receiver merely requests retransmission of the "hole" (the missing ADU) in the flow of ADUs. No action is taken by the error control when $R^i_{loss} \le P^i_{loss}$.

The retransmission strategy for the proposed application protocol is based on the actual MU loss rate (R^i_{loss}) and the desired QoS. It is different from the time-based error recovery schemes such as those in MSTP [18], LVMR [19], and RVTR [20]. The error control process buffers ADUs for in-sequence checking and the optional checksum calculation, so it only introduces a small delay of processing on ADUs, which is insignificant in comparing with the end-to-end delay.

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) from 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, which can be obtained from the *Global Positioning System* (GPS). 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. 5.

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 new value of R^i_{late} is calculated. The calculation of R^i_{late} is to divide the number of late MUs by the total number of MUs that should be received. Similarly as the error control, if $R^i_{late} > P^i_{late}$, which means the QoS is violated, the connection should be terminated or enter the re-negotiation state. The MUs that are reassembled within the end-to-end delay bound are passed immediately to the next control mechanism, i.e. the synchronization control.

4.3 Synchronization control

The only function of the synchronization control is to synchronize the delivery of all MUs with the same *Timestamp*. Therefore, the process latches the MUs with the same *Timestamp* until the MUs of all connections arrive or the local time (*Timestamp + D*) is up. In other words, all MUs with the same *Timestamp* must be synchronously delivered to the data consumption module within the delay bound (*D*). The process for the synchronization control is illustrated in Fig. 6.

Notice that the buffering time of one MU in the real-time control and the synchronization control plus the actual end-to-end delay of the MU does not exceed the end-to-end delay bound. However, not all MUs with the same *timestamp* are delivered

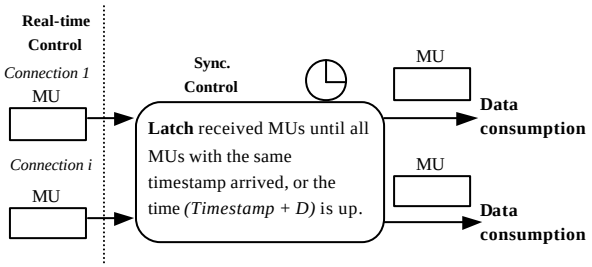


Fig. 6. The synchronization control mechanism

since some of them are probably late and are dropped in the process of the real-time control.

4.4 Quality of the synchronized session

The quality of each connection could be quantified in terms of the QoS parameters. We define the quality of a connection, P^i_{succ}, to be the percentage of successful MUs that pass all the control mechanisms. The following equation could be easily derived:

$$P^i_{succ} = 1 - (R^i_{loss} + R^i_{late}) \quad (1)$$

Under the steady state of the connection, we have $R^i_{loss} \leq MaxP^i_{loss}$ and $R^i_{late} \leq P^i_{late}$. Thus, we have

$$P^i_{succ} = 1 - (R^i_{loss} + R^i_{late}) \geq 1 - (MaxP^i_{loss} + P^i_{late}) \quad (2)$$

We assume that the MUs of each connection are of the same importance. The quality of the session, denoted by $Q_{session}$ could be computed as the mean of the connection quality, i.e.,

$$Q_{session} = \frac{\mathbf{Sum}_i P^i_{succ}}{\#\,connection\ s} \geq 1 - \frac{\mathbf{Sum}_i (MaxP^i_{loss} + P^i_{late})}{\#\,connection\ s} \quad (3)$$

Equation (2) and Equation (3) present the relationship of the parameters of application QoS and the quality of the session (and individual connection). Thus, the equations could be a good reference in determining the values of QoS.

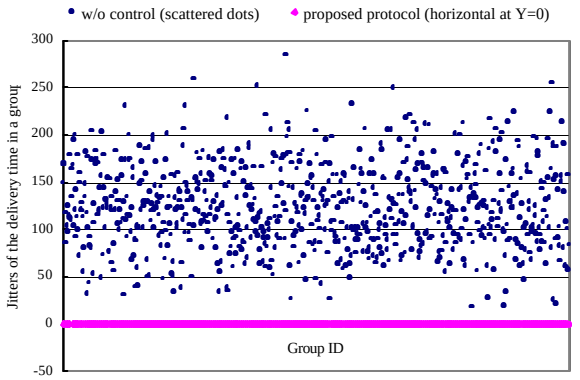
4.5 Slow-start and slow-stop

More fluctuations of the network situation occur in the beginning of the connection (session). In order to avoid a connection from frequently entering the re-negotiation state (termination and re-connection) in the beginning of the connection, the application should delay the execution of the control mechanisms for the beginning of the connection. 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 a connection is violated. The connection 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 of the connection.

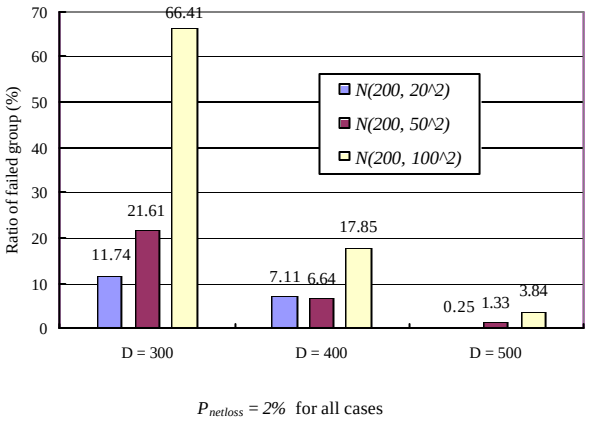
V. PERFORMANCE EVALUATION

We conducted some simulations to investigate the performance of the proposed protocol. The network environment is modeled as an ADU dropper followed by an end-to-end delay generator. The



D = 300 ms, Network: N(200, 50^2), $P_{netloss}$ = 2%

Fig. 7. Jitters of the synchronous group



$P_{netloss}$ = 2% for all cases

Fig. 8. Performance for different *D* and network delays

ADU dropper drops ADUs with probability $P_{netloss}$ and the delay generator generates end-to-end delay for each non-dropped ADU according to *Normal distribution* $N(\mu, s^2)$. The minimum value of the end-to-end delay is set to 20ms, and the delay of the control packet for triggering the retransmission of lost ADUs is also set to 20ms in the simulation.

The traffic generator regularly generates audio MUs, video MUs, and HTML MUs that are then fed into the network model and the proposed protocol. The generation rates for audio and video are both 10 MU/sec, the rate for HTML is 1 MU/2sec. MUs with the same generation time, which form a *synchronous group*, should be synchronously delivered for playback at the receiver. Moreover, one audio MU generates only one ADU, so does HTML medium. One video MU generates several ADUs. The traffic generator randomly selects the number of ADU over 3 ~ 7 for each video MU.

The application QoS parameters are set as follows:

(1) We assume the underlying network system merely provides best-effort transmission facility and no negotiation mechanism for network QoS is provided, so we set both $MaxP^i_{loss}$ and P^i_{late} to 1 for each medium such that the application cannot terminate the connections under bad network condition.

(2) Setting P_{loss}^{i} to 0 so that the retransmission of the dropped ADU is always enabled.
(3) Several values of D (delay bound) are selected in the simulation for comparison.

The jitters of a synchronous group are defined as the maximum offset of the delivery time of MUs in the same group. Fig. 7 displays the jitters for the proposed protocol as well as the contrast case of without any control (w/o control). The protocol performs real-time control and synchronization control to shape the arrival pattern to a synchronous pattern. Out-of-real-time MUs are discarded. Thus, the shaping effect of the protocol is in the cost of increasing ratio of failed synchronous group.

One the other hand, if the delay bound D is large enough so that some of retransmissions of lost ADUs could arrive in time, the performance of the protocol is improved. Fig. 8 shows performance for different D under different network delay behaviors. As shown in the figure, larger D results in better performance, and smaller variation of network delay results in better performance as well.

Notice that the performance of $N(200, 20^2)$ for $D=400ms$ is worse than that of $N(200, 50^2)$. The reason is: for $N(200, 20^2)$, once an ADU is lost, the chance for the retransmission to arrive in time is less than that of $N(200, 50^2)$, since $N(200, 20^2)$ usually generates delay close to $200ms$ and total elapsed time for the retransmission arriving at the receiver is always larger than 400ms (delay bound). On the other hand, $N(200, 50^2)$ which has a larger variation could have more chance to generate delay much less than $200ms$ and the total elapsed time sometime is less than the delay bound $400ms$.

VI. CONCLUSION

Combining the concept of ALF with the network-awareness, an application-level protocol for the network applications with the synchronized multimedia session was proposed in the paper. The application model and the parameters of the application QoS were also proposed. The control mechanisms, including the error control, the real-time control, and the synchronization control, for supporting the application-level protocol and QoS were presented. Moreover, the quantitative expression for the quality of the synchronized session was derived in the paper and could be used in determining the values of QoS parameters for the session. Simulation results show the effectiveness of the protocol. Comparisons for different values of the delay bound under different network condition are also presented in the paper.

REFERENCES

[1] A. Vogel, B. Kerherve, G. Bochmann, and J. Gecsei, "Distributed Multimedia and QoS: a survey," IEEE Multimedia, vol. 2, no. 2, pp. 10-19, summer, 1995.

[2] P. G. S. Florissi, Y. Yemini, and D. Florissi, "QoSockets: a New Extension to the Sockets API for End-to-End Application QoS Management," Proceedings, 6th IFIP/IEEE International Symposium on Integrated Network Management," 1999, pp. 655-668.

[3] R. Braden, L. Zhang, D. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP)- Version 1 functional specification. Internet Draft, Internet Engineering Task Force, 1996.

[4] P. Y. Wang, Y. Yemini, D. Florissi, J. Zinky, and P. Florissi, "Experimental QoS Performance of Multimedia Applications," Proceedings, IEEE 19th annual joint conference of the IEEE Computer and Communications Societies, 2000 (INFOCOM 2000), pp. 970-979.

[5] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy, "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation," Proceedings, 7th IEEE International Workshop on Quality of Service (IWQoS'99), 1999, pp. 27-36.

[6] A. G. Malamos, E. N. Malamas, T. A. Varvarigou, and S. R. Ahuja, "On the Definition, Modeling, and Implementation of Quality of Service (QoS) in Distributed Multimedia Systems," Proceedings, IEEE International Symposium on Computers and Communications, 1999, pp. 39-403.

[7] S. Chatterjee, J. Sydir, B. Sabata, and T. Lawrence, "Modeling Applications for Adaptive QoS-based Resource Management," Proceedings, High-Assurance Systems Engineering Workshop, 1997, pp. 194-201.

[8] T. F. Abdelzaher and K. G. Shin, "QoS Provisioning with qContracts in Web and Multimedia Servers," Proceedings, 20th IEEE Real-Time Systems Symposium, 1999, pp. 44-53.

[9] S. Brandt, G. Nutt, T. Berk, and J. Mankovich, "A Dynamic Quality of Service Middleware Agent for Mediating Application Resource Usage," Proceedings, 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," Proceedings, IEEE International Conference on Communications (ICC' 99), 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," Proceedings, IEEE Wireless Communications and Networking Conference (WCNC), 1999, pp. 368-372.

[12] D. Clark and D. Tenenhouse, "Architectural Considerations for a New Generation of Protocols," ACM SIGCOMM' 90, pp. 200-208.

[13] H. Schulzrinne, S. Casner, R.Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Appli-cations," RFC1889, Nov. 1995.

[14] J. Bolliger and Th. Gross, "A Framework-based Approach to the Development of Network-Aware Applications," IEEE trans. Software Engineering, vol. 24, no. 5, May 1998, pp. 376-390.

[15] J. Bolliger and Th. Gross, "Bandwidth Modelling for Network-Aware Applications," Proceedings., IEEE INFOCOM' 99, 1999, pp. 1300-1309.

[16] K. Lai and M. Baker, "Measuring Bandwidth," Proceedings, IEEE INFOCOM' 99, 1999, pp. 235-245.

[17] Vern Paxson, "End-to-End Internet Packet Dynamics," IEEE/ACM trans. Networking, vol. 7, no. 3, June 1999, pp. 277-292.

[18] Chun-Chuan Yang and Jau-Hsiung Huang, "A Multimedia Synchronization Model and its Implementation in Transport protocols," IEEE Journal of Selected Area in Communications, vol. 14, No. 1, pp. 212-225, Jan 1996.

[19] X. Li, S. Paul, and M. Ammar, "Layered Video Multicast with Retransmission (LVMR): Evaluation of Error Recovery Scheme," Proceedings, IEEE 7th International Workshop on Network and Operating System Support for Digital Audio and Video, 1997 (NOSSDAV'97), pp. 161-171.

[20] S. Tasaka, T. Nunome, and Y. Ishibashi, "Live Media Synchronization Quality of a Retransmission-based Error Recovery Scheme," Proceedings, IEEE ICC 2000, pp. 1535-1541.