An Improved Timer Scheme of Multiple Retransmissions for Media Streaming

Hsiao-Yi Chen, Chia-Hui Wang
Department of Computer Science and Information Engineering
Ming Chuan University, Taiwan
{s3450193@ss24, wangch}.mcu.edu.tw

ABSTRACT

Sender's retransmission for the lost packet to receiver is a popular error control scheme in public network due to its cost effectiveness. However, either the ARQ(Automatic Repeat reQuest) or the retransmitted packet may be lost again. Multiple retransmissions for a lost packet are possible while the receiver's processing deadline of the lost packet is larger than the network round trip delay (i.e. RTD).

In this paper, we improve the traditional timer scheme[#] to effectively detect the loss of ARQ or retransmitted packet to furnish the multiple retransmissions. Moreover, we demonstrate that our improved timer scheme for multiple retransmissions can effectively reduce the packet loss ratio by the performance results from not only simulations, but also a True VOD system.

1: INTRODUCTIONS

In the prevalent public network of Internet, the packet loss is unavoidable and it will degrade the service quality of network applications. Therefore, we need an effective error control for network applications to preserve the service. The retransmission-based (i.e. ARQ) error control is very effective and low-cost for burst packet loss, but it need a network RTD (Round Trip Delay) to complete its error control. Therefore, it was not very popular in time-critical real-time applications such as interactive multimedia communications.

However, retransmission-based error control is still very economical and attractive error control while the process deadline of lost packet is longer than the network round trip delay. As shown in Figure 1, receiver can use gap-based loss detection [1] to immediately detect the loss of packet *i* in the meantime of receiving the packet *i*+1 and sends the ARQ to sender to retransmit packet *i*. However, the retransmitted packet *i* could be lost (or even the ARQ for packet *i* may be lost) in the public network. Then, we need a timer-based loss detection to detect the loss of retransmitted packet or the ARQ and then we can issue another ARQ to ask sender to send again the lost packet *i*. This scheme is so-called **multiple retransmissions** to endeavor to retransmit the lost packet if the network RTD is allowed.

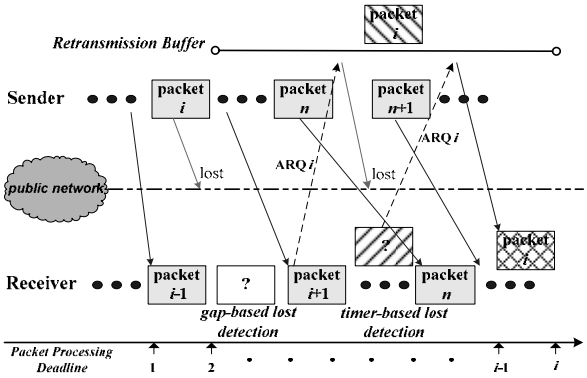

Figure 1. Retransmission-based error control scheme.

Due to the uncertainty in public network, packet loss and RTD may vary all the time. We need an effective timer-based loss detection to help multiple retransmissions to recover as many lost packets as possible.

In this paper, we propose an improved timer scheme for multiple retransmissions in retransmission-based error control. Not only the time complexity to either start or stop a timer to achieve multiple retransmission is O(1), but also the ratio of packet loss can be significantly reduced due to the effective multiple retransmissions.

In the following section, we present an analytic model of multiple retransmissions about the theoretic performance. In section 3, conventional timer of callout queue and our improved scheme are introduced. In section 4, we apply our improved timer scheme into a true VOD (Video On Demand) system to demonstrate the enhanced playback quality of service by experimental results. In the final section, conclusions and future work are presented.

2: ANALYTIC PERFORMANCE OF MULTIPLE RETRANSMISSIONS.

While the process deadline of lost packet is much larger than the RTD, multiple retransmissions for a loss packet is feasible because of retransmitted packet and ARQ may be lost in the public network. Then, we assume that the loss ratio of the uplink of ARQ is ***u*** and the loss ratio of the downlink of retransmitted packet is ***d***.

[#] The work was partially done while the author was visiting the Institute of Information Science, Academia Sinica, Taiwan in 2005 and supported by National Science Council, Project No. NSC94-2213-E-130-001, Taiwan.

After sender delivers ***p*** packets to receiver, the number of lost packets will be *pd*.

While we apply retransmission-based error control and the round trip delay is shorter than the packet's processing deadline (PPD) of lost packets, the receiver will send *pd* ARQs at maximum to ask sender to retransmit lost packets. Due to the loss ratio of *u* in the uplink, sender will receive *pd*(*1-u*) ARQs and then retransmit *pd*(*1-u*) lost packets from retransmission buffer to receiver. However, because of the loss ratio of *d* in the downlink, $pd^2(1-u)$ retransmitted packets will be lost again. After first attempt of retransmissions for lost packets, previous *pd* (denoted as L_0) lost packets are reduced to L_1, L_1 is expressed as follows:

$$L_1 = L_0 u + L_0 d(1-u) = L_0[u + d(1-u)] \quad (1)$$

Moreover, if the second attempt of retransmission for L_1 lost packet is possible, the lost packets can be reduced to L_2:

$$L_2 = L_1 u + L_1 d(1-u) = L_0[u + d(1-u)]^2 \quad (2)$$

Therefore, while the multiple retransmissions successfully retransmit the lost packets from *p* delivered packets in order *n* (i.e. $n = \lfloor PPD/RTD \rfloor$), the reduced loss ratio of delivering *p* packets is:

$$\frac{L_n}{p} = d(u + d - ud)^n \quad (3)$$

Considering the traffic of retransmitted packets of multiple retransmissions in order *n*, the total contributed traffic to the downlink to retransmit the lost packet is:

$$p\sum_{i=1}^{n}[d(1-u)]^i = pd(1-u)\frac{1-[d(1-u)]^n}{1-d(1-u)} \quad (4)$$

Moreover, if a network application delivers its data to downlink in a bit rate of *r*, we simply extend the formula (4) to indicate the extra bit rate needed to complete the multiple retransmissions in order *n*. The extra bit rate is shown as follows:

$$rd(1-u)\frac{1-[d(1-u)]^n}{1-d(1-u)} < r\frac{d(1-u)}{1-d(1-u)} \quad (5)$$

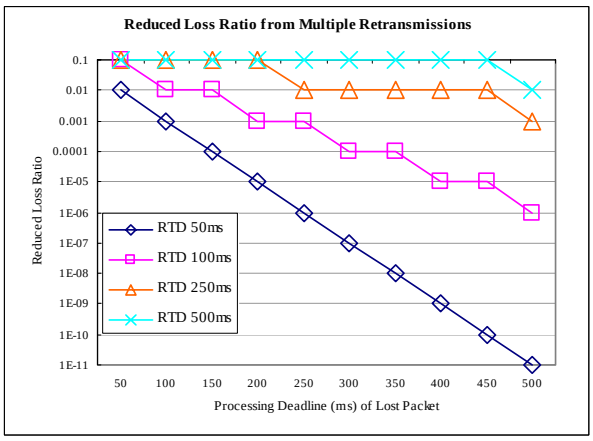

Figure 2. Reduced loss ratio from multiple retransmissions.

For an example of the reduced loss ratio from multiple retransmissions as shown in Figure 2, the uplink loss ratio is 0, the downlink loss ratio is 0.1, RTD ranges from 50 to 500ms and PPD ranges from 50ms to 500ms. Multiple retransmissions can significantly reduce the packet loss ratio if the PPD is much shorted than RTD. According to the formula (4), the extra traffic for multiple retransmissions is less than 1/9 of the original data traffic.

From the analytic performance of multiple retransmissions presented in previous section, we believe that the improved ratio for multiple retransmissions is exponential. To furnish multiple retransmissions, we need an effective timer scheme to effectively detect whether the ARQ or the retransmitted packet is lost in a network RTD.

3: IMPROVED TIMER SCHEME FOR MULTIPLE RETRANSMISSIONS

To effectively detect the loss of ARQ or retransmitted packet, multiple retransmissions need to register a timeout event with the timeout value of network RTD while we issue an ARQ for lost packet. If the retransmitted packet arrives at receiver before the timeout, the receiver's timer scheme in multiple retransmissions must remove the timeout event for this successfully retransmitted packet.

However, if a timeout event for lost packet reaches the timeout value and the retransmitted packet has not arrived yet, we believe that the ARQ or the retransmitted packet was lost during the previous period of RTD. Then, multiple retransmissions must immediately decide to issue another ARQ to sender again and reschedule another timeout event in the meantime, if the packet processing deadline is still larger than the RTD. Otherwise, multiple retransmissions must remove the timeout event to abandon next retransmission for this lost packet, because this lost packet is impossible to be recovered and has no need to be retransmitted again to waste the network traffic. This is so-called **conditional retransmission** [1] to prevent redundant retransmission, multiple retransmission issue ARQ to sender to retransmit lost packet only while the network RTD is allowed.

3.1: TIMER ACTIONS IN MULTIPLE RETRANSMISSIONS

Therefore, there are four kinds of actions related to the timer to achieve multiple retransmissions (MR). The first one is to add a new timeout event for a lost packet, which was detected by receiver's gap-based loss detection. The abbreviation of this action is given as *newMRtimer*(*seq*, *rt*). The parameter *seq* in the function *newMRtimer* stands for the sequence number of lost packet. The *rt* stands for the timeout value of RTD.

The second one is to remove a timeout event while a correspondent retransmitted packet is received before the timeout. This action is abbreviated as *removeMRtimerEarly*(*seq*, *rt*). The third one is to remove the timeout event while no retransmitted packet

arrives till the timer for the lost packet reaches its timeout and the next retransmission cannot be rescheduled due to the RTD. This third action is abbreviated as *removeMRtimerLate*(*seq*, *rt*).

If no retransmitted packet arrives till the timeout for lost packet and the network RTD is allowed to reschedule next retransmission for this lost packet, the final action is to issue another ARQ and then to renew the timer for the lost packet. The final action is abbreviated as *renewMRtimer*(*seq*, *rt*).

Now, we give a brief discussion about the processing complexity for these four actions while multiple retransmissions apply the well-known callout queue [4] timer scheme. The original callout queue maintains a link list of timeout events. In callout queue, each node of timeout event has relative timeout value to its previous node. Therefore, the bookkeeping (i.e. so-called PERTICKBOOKKEEPING in [6]) to check any expired timer events only needs to decrease one from the timeout value in the first node of callout queue. While the first node's timeout value reaches to zero, the first node have to be removed from the callout queue and its timeout event should be triggered. The processing complexity (i.e. PERTICKBOOKKEEPING) to trigger a timeout event is O(1). However, due to the sorted structure in callout queue, to insert a timeout event is time complexity O(*n*).

While we apply the callout queue scheme in multiple retransmissions, we can simply estimate that the time complexities of *newMRtimer*, *removeMRtimerEarly*, *removeMRtimerLate* and *renewMRtimer* actions are O(*n*), O(*n*), O(1) and O(*n*) respectively.

3.2: IMPROVED TIME COMPLEXITY FOR TIMER ACTIONS

Due to our two observations about the insertion actions of *newMRtimer* and *renewMRtimer* and deletion actions of *removeMRtimerEarly*, we believe that the time complexities of these three actions can be improved further. The first observation is that all the timeout values of timeout events for previous lost packets in callout queue should be adjusted while the current observed RTD is different with previous RTD. It's because those previous lost packets' timeout values should be consistent with the latest observed RTD. Therefore, the timer-based loss detection won't be inaccurate to retransmit another ARQ for multiple retransmissions, due to the variation of network RTD.

According to the first observation, the timer nodes of the actions of *newMRtimer* and *renewMRtimer* should be always inserted to the tail of callout queue. It's because that the all the previous timeout events must be trigger earlier than the newly-added timer node even though the newly-added node preserves smaller RTD. Therefore, the time complexity of *newMRtimer* and *renewMRtime* can be O(1).

However, while the current observed RTD is different with previous one, all the timeout values of timeout events for lost packets must be adjusted effectively. Our proposed method is to add a storage called "*RTD Diff*" to help the modified callout queue scheme for multiple retransmissions to update all the timeout values effectively. As shown in Figure 3, *RTD Diff* will store the value of the difference between the current RTD and previous RTD (i.e. the previous RTD minus the current RTD).

$$rtd\_diff = previous\_rtd - current\_rtd \quad (6)$$

While the RTD is varied, the *RTD Diff* will have the value of *rtd_diff* as shown in formula (6). Then, *RTD Diff* will be reset to zero after PERTICKBOOKKEEPING decrease one and *rtd_diff* from the timeout value in the first node of callout queue. The time complexity of PERTICKBOOKKEEPING is still O(1) even if the network RTD is varied all the time.

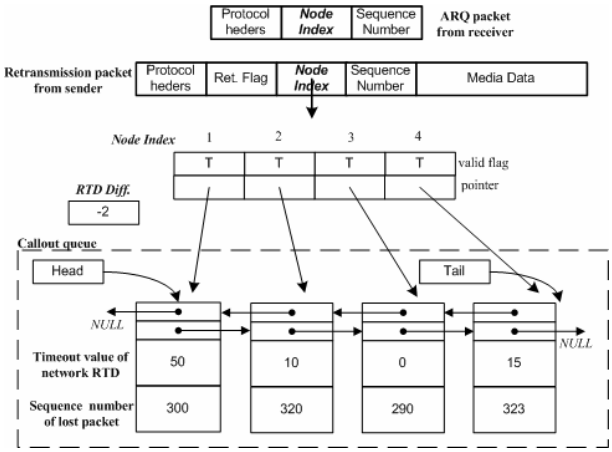

Figure 3. A sample instance of callout queue timer scheme applied in multiple retransmissions.

Besides, the retransmitted packet may arrive at the receiver before its timeout in callout queue. This retransmitted packet's timeout event should be immediately removed from callout queue. As shown in Figure 3, our proposed method to effectively furnish the action of *removeMRtimerEarly* is to store the additional information of *node index* into the ARQ packet. Usually, the ARQ packet from receiver only contains the lost packet's sequence number to ask sender to retransmit the lost packet with the correspondent sequence number. Then, sender will directly retransmit the packet in retransmission buffer according to the sequence number. The additional information *node index* is the index to an array of addresses for all timer nodes in callout queue including the validation flag for the timer node.

While sender receives the ARQ packet with *node index* and sequence number, sender will also copy the *node index* and sequence number into retransmitted packet's header. Then, after receiver receives the retransmitted packet, our modified callout queue timer scheme can directly remove the timer node without searching. It's because not only the explicit address information of removed node indicated by the *node index*, but also the callout queue has been modified to

be a double link list. The time complexity of *removeMRtimerEarly* is reduced to O(1). Therefore, all the time complexities of *newMRtimer*, *removeMRtimerEarly*, *removeMRtimerLate* and *renewMRtimer* actions are O(1).

4: SIMULATIONS, VOD EXPERIMENTS AND PERFORMANCE RESULTS

To validate the performance of our proposed improved timer scheme of multiple retransmissions for media streaming, not only the simulations are conducted to demonstrate the performance of the improved timer scheme, but also the experiments are conducted for a true VOD system to demonstrate that the multiple retransmissions, which apply the modified timer scheme, can significantly reduce the packet loss and improved the QoS of VOD system.

4.1: SIMULATIONS AND PERFORMANCE RESULTS FOR IMPROVED TIMER SCHEME

Our simulations to demonstrate the performance of callout queue are conducted by invoking 500 threads at different timeout intervals and every thread will hook a timer whose timeout value is averagely 100ms with different probability distributions.

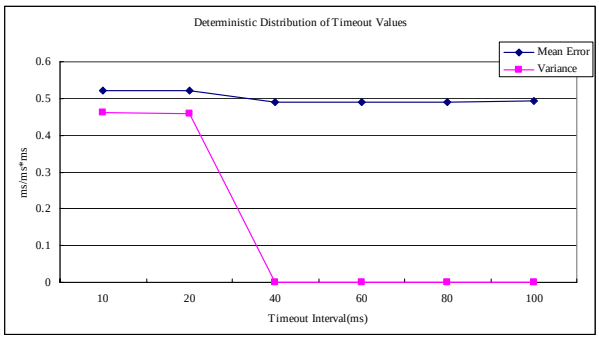

Figure 4. Mean error and variance of deterministic distribution of timeout values at different timeout interval.

The timeout intervals to issue a new thread to hook a timeout event ranges from 10ms to 100ms. The probability distributions of the timeout value 100ms are deterministic, exponential and general respectively. We record the actual start time and stop time for each thread to obtain the actual timeout value. Then we calculate the error and variance between the hooked timeout value and actual timeout value to validate the performance of proposed timeout scheme for multiple retransmissions. These simulations results are shown in Figure 4, 5 and 6 respectively,

According to the results shown in the figures, all the mean errors are much less than 1ms. We can say that the variation of timeout value won't affect the mean error. However, the length of timeout interval will affect the variance. It indicates that a lot of lost packets (such as burst packet loss) to hook the timeout events at the same time. Then, the processing overhead to trigger many timeout events at the same time will affect the accuracy of the callout queue timer. We can also see that the mean errors are slightly smaller while the timeout interval getting longer.

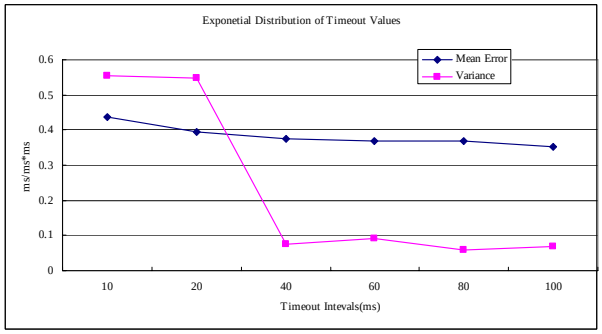

Figure 5. Mean error and variance of exponential distribution of timeout values at different timeout interval.

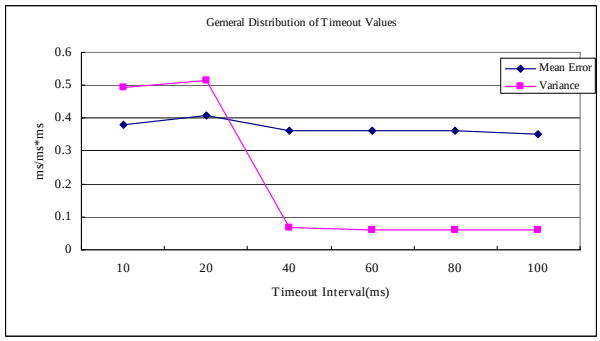

Figure 6. Mean error and variance of general distribution of timeout values at different timeout interval.

4.2: VOD EXPERIMENTS AND PERFORMANCE RESULTS FOR MULTIPLE RETRANSMISSIONS

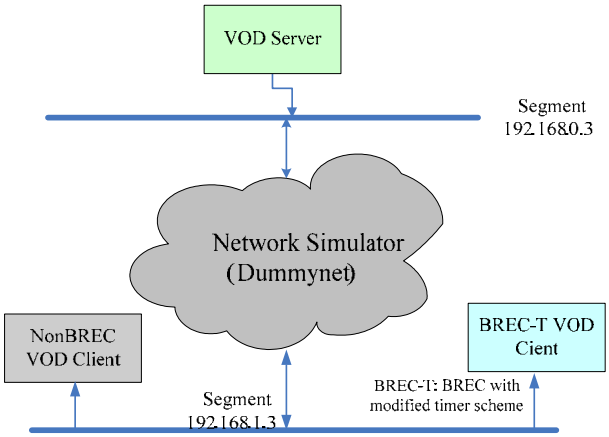

Figure 7. Experimental test-bed for BREC VOD with modified timer scheme.

We apply our modified timer scheme into a VOD system, which is implemented by JMF (Java Media Framework) and supports BREC (Buffer-controlled Retransmission Error Control) scheme [7] to demonstrate the performance of multiple retransmissions. The BREC scheme applies the feedback control of receiver's buffer occupancy to

dynamically adjust sender's sending rate. While the receiver's buffer occupancy is controlled at a given threshold, it indicates that PPD has been extended and the multiple retransmissions can be applied successfully.

Furthermore, previous BREC scheme with modifier timer scheme (abbreviated as BREC-T) is applied into a true VOD system. This VOD system is design by Java Media Framework (JMF). The performance of multiple retransmissions is examined in a test-bed as shown in Figure 7. We use Dummynet [9] to simulate the network uncertainty of delay, loss and bandwidth. We have two different types of VOD clients, the first one is BREC-T client and the second one is NonBREC client.

In this paper, we only demonstrate the experimental results of BREC-T, because the performance comparisons between BREC and NonBREC were presented in [7]. Besides, the applied rate control function has been changed to formula (7) as shown below rather than the *P* and *PD* rate control functions applied in BREC [7].

$$r(t) = mean\_data\_rate + \Delta r_M * (b_m - b(t))/b_m \quad (7)$$

$r(t)$ is the sender's sending rate and *mean_data_rate* is the average data of media content. For example, the *mean_data_rate* of MPEG-1 media is 1.5Mbps. Δr_M is the maximum rate adjustment to avoid large burst traffic flooding to the network. $b(t)$ represents the current buffer occupancy in the feedback control and b_m is the target position of buffer occupancy.

Usually, b_m is the middle level in the playback buffer and Δr_M is as high as 15KBps (quite close to the criteria in formula (5) if the maximum loss rate is 0.1) in our experiments. The receiver's playback buffer can accommodate 500 packets (i.e. b_m =250) and each packet size is about 1KB. The $r(t)$ is also limited to 4Mbps in our experiment to limit the maximum traffic for a single VOD service connection. The feedback control intervals conducted in our experiments are 50ms, 250ms and 500ms.

In our experiments, we also examine 3 different kinds of MPEG-1 videos in our test cases, the first one is Star War-Episode I with a lot of scene changes and motions, the second one is Weatherman with less scene changes and motions and the last one is Football. Then, we apply 3 different kinds of RTD and loss rates by using Dummynet. The different RTDs are 20ms, 500ms and 1000ms. The different network loss rates are 0.01, 0.05 and 0.1. The feedback control intervals conducted in our experiments are 50ms, 250ms and 500ms. The consecutive 40000 packets are recorded in each test case of the experiments.

The controllability of our BREC-T is shown in Figure 8. Compared to the previous BREC [7], the BREC-T has better controllability even sustaining the network loss rate 0.01 and different RTDs. But, the feedback control conducted in BREC-T experiments is 500ms and it's slightly smaller than the feedback interval applied in BREC.

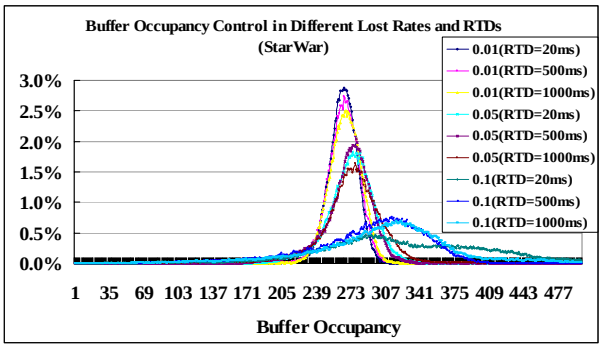

Figure 8. Controllability of buffer occupancy in different RTDs.

After studying the experimental results in Figure 8, we believe that the loss ratio will seriously impact the buffer controllability. RTD won't affect the buffer controllability too much. That's because when the receiver's media decoder eventually detects an error in received data, the decoder will skip a portion of data to next readable data boundary. At the mean time, many packets in playback buffer will be discarded immediately. It means the decoding rate of arrival packets in playback buffer will get much bigger than ever. That's why the network loss ratio will seriously impact the buffer controllability.

The controlled buffer occupancy also indicates that most of the packet's PPD is extended to a limit without affecting the playback QoS and the multiple retransmissions can be effectively achieved.

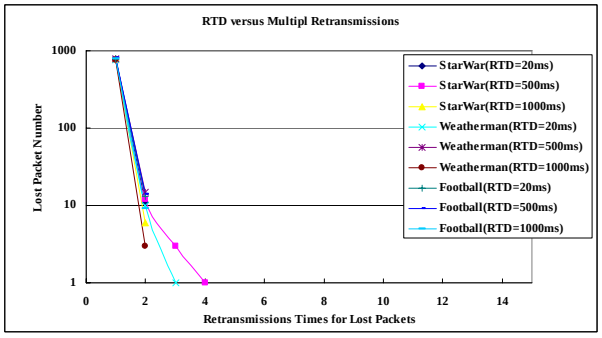

Figure 9. Retransmission times for lost packets at the network loss ratio 0.01.

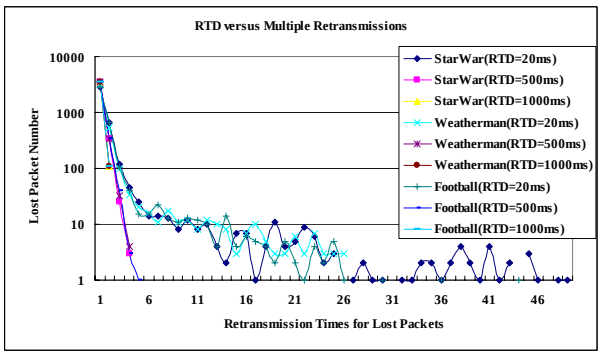

Figure 10. Retransmission times for lost packets at the network loss ratio 0.05.

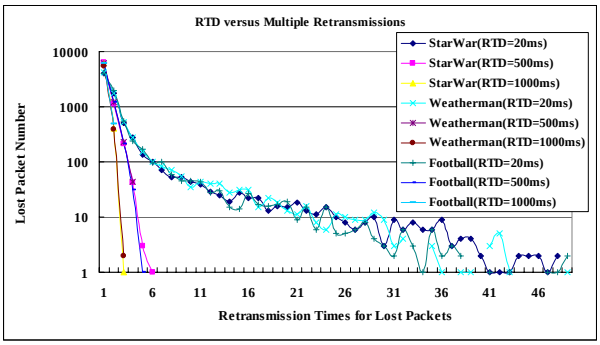

Figure 11. Retransmission times for lost packets at the network loss ratio 0.1.

Now, we examine the retransmission times of multiple retransmissions in BREC-T to validate if the packet loss rate can be effectively reduced. In the experiments, we recorded the retransmission times for each loss packet while applying 3 different network loss rates, 3 different network RTDs and 3 different types of movies mentioned above. The experimental results are shown in Figure 9, 10 and 11 respectively.

As shown in Figure 9, the retransmission times for lost packets are very small while the inserted network loss ratio is small. It's because that most of the lost packets are successfully retransmitted to receiver at the first time due to the small network loss ratio. However, the network RTD seriously affects the retransmission times for multiple retransmissions. While the network loss rate is high enough and the RTD is small, the retransmission times for lost packets will be large.

According to our analytic model in Section 2, the observed loss ratio can be significantly reduced while high order of multiple retransmissions can be achieved. As shown in Table 1, the experimental results of observed loss ratio from the VOD system with BREC-T scheme also demonstrate that the multiple retransmissions with improved timer scheme can enhance the QoS of the VOD system.

Table 1. Observed loss rates for different times of multiple retransmissions.

Ret. times / RTD	0	1	2	>2
10ms	0.01	0.077	0.05524	0.00088
500 ms	0.01	0.098	0.074	0.00094
1000 ms	0.01	0.10014	0.0839	0.0182

5: CONCLUSIONS AND FUTURE WORKS

In this paper, we modify the callout queue timer scheme for multiple retransmissions. All the correspondent actions in callout queue are time complexities of O(1). Not only the simulations demonstrate the effectiveness of the modified timer scheme, but also we also deploy multiple retransmissions with this timer scheme into a true VOD system to demonstrate that the multiple retransmissions can improve the QoS of VOD due to the significantly reduced packet loss ratio. Through the theoretic analysis, simulations and experimental results, we believe that BREC-T error control scheme can effectively enhance the playback quality for video streaming.

In the near future, we will apply our BREC-T into a mobile network environment to validate the QoS improvement of video streaming, because the ARQ usually preserves better performance than other error control schemes in handling burst packet loss while the mobile receiver handoff to another wireless access network. We will also examine our BREC-T by applying other streaming media such as famous MPEG-4.

REFERENCES

[1] C. Papadoulos and G. Parulkar, "Retransmission-based error control for continuous media applications," in Proc. 6th Int. Workshop Network Operating Syst. Support Digital and Audio Video, 1996, pp.5-12.

[2] G. Carle, E. W. Biersack, "Survey of Error Recovery Techniques for IP-based Audio-Visual Multicast Applications," IEEE Network, Vol.11, no. 6, pp.24-36, Nov. 1997.

[3] J. Bolot, T. Turletti, "A rate control mechanism for packet video in the Internet," INFOCOM '94. Networking for Global Communications., 13th Proceedings IEEE, 1994 Page(s): 1216 -1223 vol.3.

[4] A. Costello and G. Varghese, "Redesigning the BSD callout and timeout facilities," Dept. Computer Science, Washington Univ., St. Louis, MO, Tech. Rep. 95-23, Sept. 1995.

[5] C. Wang, J. Ho, S. Hsu, et.al., "Design and Implementation of an Interactive True VOD System on ADSL with Scarce Resources at the Set-top Box," International Computer Symposium on Workshop on Computer Networks, Internet, and Multimedia, Taiwan, 2000.

[6] George Varghese and Anthony Lauck, "Hashed and Hierarchical Timing Wheels: Efficient Data Structures for Implementing a Timer Facility," IEEE/ACM Transactions On Networking, Vol. 5, No.6, Dec. 1997.

[7] Chia-Hui Wang, Ray-I Chang, Jan-Ming Ho, and Shun-Chin Sheu, "Rate-Sensitive ARQ For Real-Time Video Streaming," IEEE Globecom, pp.3361–3365, 2003.

[8] Rishi Sinha, Christos Papadopoulos, "Streaming: An adaptive multiple retransmission technique for continuous media streams," Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video, June 2004.

[9] Dummynet, http://info.iet.unipi.it/~luigi/ip_dummynet/