FRames-Ensured-Delivery Algorithm for Smoothing and Multiplexing Prerecorded Videos

Yaakov A. Bembaron
Department of Electrical and Computer Engineering
Ben-Gurion University of the Negev
Beer-Sheva , 84105 Israel
bembaron@bgumail.bgu.ac.il

Ofer Hadar
Department of Communication Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva , 84105 Israel
hadar@bgumail.bgu.ac.il

Abstract **Video-on-Demand (VoD) Servers normally store MPEG compressed versions of prerecorded videos in order to reduce their network bandwidth demands. Compressed videos have variable-bit-rate (VBR) for a specific compression quality. Transmission of videos on the network at a peak rate results in poor bandwidth utilization, while transmission at a lower rate results in quality-of-service (QoS) degradation. Implementation of a traffic smoothing on the VBR stream, using a buffer at the client receiver, is a widely known approach to overcome the rate variability problem. This paper presents a novel smoothing and multiplexing protocol for prerecorded videos that constructs a FRames-Ensured-Delivery (FRED) and constant-bit-rate (CBR) transmission plan. The protocol achieves perfect QoS, and also provides ideal bandwidth utilization without any frame loss. According to the proposed protocol, the server rate can be changed to explore its affect on the startup latency and the client's buffer size requirements. Simulation results are provided. We compare FRED protocol to a published work called Join-the-Shortest-Queue (JSQ). FRED protocol overcomes JSQ protocol in places where there is a lack of bandwidth or where the statistical multiplexing gain is not big enough.**

Index Terms — Video-on-Demand, FRames-Ensured-Delivery, prerecorded video, Join-the-Shortest-Queue, Variable-Bit-Rate, bandwidth utilization, statistical multiplexing, Quality-of-Service, traffic smoothing, multiplexed video, Constant-Bit-Rate

I. INTRODUCTION

Digital networks technology has made rapid strides over the last few years. Nowadays, digital videos services are applicable for most ADSL or CABLE internet subscribers. The commercial market is full with digital video applications such as E-learn, VoD libraries, Near-VoD libraries and more. Video servers normally store MPEG compressed versions of prerecorded videos in order to reduce their storage and network bandwidth demands. Compressed videos have variable-bit-rate (VBR) characteristic for a specific compression quality. Transmission of videos on the network at peak rate results in poor bandwidth utilization, while transmission at a lower rate results in frame losses and Quality-of-Service (QoS) degradation. A widely known approach to solve this problem is to implement a traffic smoothing algorithm using a buffer to reduce server rate variability.

Many smoothing methods have been proposed for a single video [1]-[10]. All of these methods optimize one parameter or more for an individual video. However, in the case of several multiplexed streams, these methods result a sub-optimal transmission plan because they are not intended for smoothing a multiplexed stream and disregard statistical multiplexing gain in their algorithm. Numerous smoothing protocols for multiplexed videos have been proposed in the literature [11]-[17]. While these protocols enjoy the benefits of statistical multiplexing gain, they do not provide a lossless rate smoothing and a Constant-Bit-Rate (CBR) transmission plan at the same time, therefore they do not utilize the server bandwidth and also preserve the QoS.

This paper presents a novel smoothing and multiplexing protocol for prerecorded videos, which constructs FRames-Ensured-Delivery (FRED) and Constant-Bit-Rate (CBR) transmission plan. The proposed FRED protocol achieves perfect QoS without any frame loss and also provides ideal bandwidth utilization using a smoothed CBR transmission plan. The tradeoffs for this perfect plan are larger client's buffer size requirements and larger startup latency for initial prefetching. FRED protocol uses a-priori information about future frames sizes, and thus succeeds in allocating the entire server bandwidth among its clients, so each client has enough data to present its frames on time. The proposed protocol promises a full use of server bandwidth, by using the rest of the bandwidth for prefetching. The prefetching policy objective is to equalize the prefetched data among all clients, so clients buffer size requirements are as low as possible. Initial prefetching is performed to avoid lack of data on the clients which results in loss of frames and QoS degradation. The server CBR can be adjusted as a parameter to FRED protocol, changing affects the initial latency and the client buffer size requirements.

The paper is organized as follow. Section II presents and describes the pseudocode of the FRED algorithm. Section III compares the FRED protocol to previously published smoothing scheme for multiplexed video stream (JSQ [11]). Section IV draws some conclusions.

II. FRED ALGORITHM

There are two phases to the FRED algorithm. During the first phase the algorithm scans the videos frames from the last frame to the first frame. A CBR transmission plan is calculated to prevent frame losses. The only drawback at this phase is that there is no limitation to the amount of buffer prefetching thus in some cases, mostly at the end of a movie, the buffer occupancy can exceed the remaining amount of video to be transmitted. This situation implies that prefetching has been done for a non-existent future video stream. This extra prefetching causes bandwidth waste, degrading the bandwidth utilization.

The second phase remedies this problem. To do this first, let us denote the remaining amounts of video to be transmitted to client j starting at time t as $ACC_j(t)$. During the second phase the videos frames are scanned in chronological order. If an exception is discovered (i.e. occupancy of buffer j is larger than $ACC_j(t)$), then the buffer occupancy is lowered to $ACC_j(t)$. The spare bandwidth released from this action is then splitted equally among the other clients.

These two phases use only the buffer's occupancy matrix. Translating this matrix to rates is done by the following operator:

$$R_j(t) = Buf_j(t+1) - Buf_j(t) + FL_j(t) \qquad (1)$$

where $R_j(t)$ is the transmission rate at time t to client j, $Buf_j(t)$ is the buffer occupancy of client j at time t, and $FL_j(t)$ is the size of the frame, to be presented on client j's monitor at time t.

The FRED pseudocode is divided into four sub-sections. The first sub-section contains general notations. The second sub-section describes the first phase. The third sub-section describes the second phase, and the fourth sub-section combines all the previous sections.

Notations:

α	- Rate coefficient, should be equal to 1 usually.
J	- Number of clients.
N	- Number of frames in a movie
$FL_j(t)$	- Size of the frame to be shown on client j at time t.
$Buf_j(t)$	- Buffer occupancy of client j at time t.

The amount of data to be presented on all clients monitors, at time t is

$$FL(t) = \sum_{j=1}^{J} FL_j(t)$$

Occupancy of all clients buffers at the beginning of time t is

$$Buf(t) = \sum_{j=1}^{J} Buf_j(t)$$

Bandwidth allocated to client j at time t:

$R_j(t) = Buf_j(t+1) - Buf_j(t) + FL_j(t)$

Server total bandwidth:

$$R(t) = \sum_{j=1}^{J} R_j(t) = Buf(t+1) - Buf(t) + FL(t) = R$$

Number of bits to be shown from time t to the end of movie: $ACC_j(t) = \sum_{\tau=t}^{N} FL_j(\tau)$

END of Notations

```
Phase1( $FL_j(t)$ ,R)        {building the first transmission plan}
  $Buf_j(N+1) = 0$ ; j=1..J     {zero final buffer's occupancies}
  FOR t=N downto 1 DO            {scan frames backwards}
    {Equalize buffer's occupancies}
    $Buf_j(t) = \frac{1}{J}\left[\left(\sum_{j=1}^{J} Buf_j(t+1) + FL_j(t)\right) - R\right]$ ; j=1..J
    LOOP1
      Mark all clients as prefetchable.
      SPARE=0                    {assume no extra bandwidth}
      IF ( $R_j(t) < 0$ for any j) THEN       {if negative rate}
        SPARE=SPARE+$|R_j(t)|$  {update extra bandwidth}
        $Buf_j(t) = Buf_j(t+1) + FL_j(t)$      {make $R_j(t) = 0$ }
      Mark client j as Non-prefetchable.
      ENDIF
      IF (SPARE>0)               {if there is extra bandwidth}
      Cnt=Number of prefetchable clients.
      {Give back bandwidth of prefetchable buffers}
      $SPARE = SPARE + \sum_{j:Prefetchable} Buf_j(t+1) - Buf_j(t)$
        {Spread spare bandwidth on prefetchable buffers}
      $\underset{j:Prefetchable}{Buf_j(t)} = \frac{\left[\left(\sum_{j:Prefetchable} Buf_j(t+1)\right) - SPARE\right]}{Cnt}$ ; i=1..Cnt
      ENDIF
    IF [there was at least one negative $R_j(t)$ ] THEN
      GOTO LOOP1
    ENDIF
  ENDFOR t
  {Avoid illegal negative buffer occupancies}
  $M_j = \min_t \left[Buf_j(t)\right]$ ; j=1..J {find min buffer occupancies}
  {Make $Buf_j(t)$ non-negative}
  $Buf_j(t) = Buf_j(t) - M_j$ ; j=1..J , t=1..N+1
RETURN with $Buf_j(t)$
```

```
Phase2( $Buf_j(t)$, $ACC_j(t)$, $FL_j(t)$ )                {fixing the plan}
  Mark all clients as prefetchable.
  FOR t=1 to N DO                                 {scan frames forwards}
    LOOP2
  {Look for exceptions in buffer's occupancies}
  IF ( $Buf_j(t) > ACC_j(t)$ for any j) AND
      (Client j is Prefetchable) THEN
      SPARE=0                          {assume no extra bandwidth}
      Mark client j as non-prefetchable
      FOR τ=t to N DO                    {run till the end of movie j}
        {Update extra bandwidth}
        $SPARE = Buf_j(\tau) - ACC_j(\tau)$
        $Buf_j(\tau) = ACC_j(\tau)$                    {fix the exception}
      Cnt=Number of prefetchable clients.
        {Spread spare bandwidth on prefetchable Buffers}
      $\underset{j:Prefetchable}{Buf_j(\tau)} = \underset{j:Prefetchable}{Buf_j(\tau)} + \frac{SPARE}{Cnt}$ ; i=1..Cnt
    ENDFOR τ
    BREAK;
  ENDIF
  IF (Buffer exception has been found) THEN
    GOTO LOOP2
  ENDIF
  ENDFOR t
RETURN with $Buf_j(t)$
```

```
FRED
  Read $FL_j(t)$ from file
  $R=\alpha*Average[FL(t)]$                {calculate server bandwidth}
  $Buf_j(t)$ =Phase1( $FL_j(t)$,R)  {build first transmission plan}
  $ACC_j(t) = \sum_{\tau=t}^{N} FL_j(\tau)$               {for details, look notations}
  $Buf_j(t)$ =Phase2( $Buf_j(t)$, $ACC_j(t)$, $FL_j(t)$ )  {fix the plan}
  {Convert Buffer's occupancies to rates}
  $R_j(t) = Buf_j(t+1) - Buf_j(t) + FL_j(t)$
  DISPLAY $FL_j(t), R_j(t), Buf_j(t)$                 {Show Results}
STOP
```

III. Performances Comparison

In this section we compare the FRED protocol to a similar protocol known as JSQ. The following parameters are compared: loss probability, buffer size requirements, startup latency and processing time.

The JSQ protocol assumes a given bandwidth (i.e. R) and tries to utilize it. The algorithm calculates which one of the client buffers is the emptiest and then sends a frame to that client. JSQ repeats this procedure until there is no more bandwidth to send the desired frame. This routine repeats it self for each time frame. An extension of JSQ policy succeeds to utilize further the server bandwidth, using the fact that although the client with the emptiest buffer can't get more frames, there is still available bandwidth that can be used for other clients. This extension does not prevent the frames from spreading when it has no more bandwidth for delivering a frame to the emptiest buffer, but it does try to deliver a frame to the second emptiest buffer, and so on, until its remaining bandwidth is so low it can't deliver any frames to the clients. In the simulations presented in this section the extended version of JSQ has been used because it gives better smoothing results and was simulated in the original paper.

The simulations were written in C, and evaluated on a MATLAB workplace for convenience. The simulations ran on an Intel Pentium 4HT, 2600 MHz computer with 256 MB memory. We investigated the influence of server bandwidth and statistical multiplexing gain on two protocols: FRED and JSQ. The statistical multiplexing gain was raised by adding different MPEG-1 movies [18] to the multiplexed stream (presented on x axis). First, the FRED algorithm was tested, without restricting the client's buffer's size. The FRED algorithm has built a transmission plan, and returned the minimal client's buffers size it needed for lossless transmission plan. Then JSQ algorithm was tested using the same server bandwidth's used in FRED algorithm and using the buffer sizes delivered by FRED algorithm. Whenever JSQ did not use its entire buffer space, the minimal client buffer requirement has been lowered. The results of these simulations are presented in the following sections.

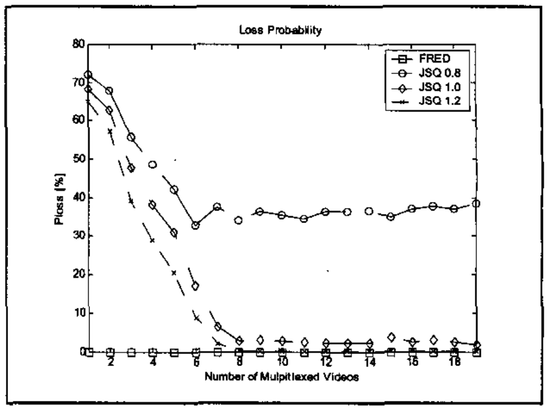

Figure 1. Comparison of Loss Probabilities of FRED and JSQ algorithms.

As can be seen in Figure 1, the FRED protocol always achieves perfect QoS, no matter what the server rate is, owing to its lossless transmission plan. Nominal bandwidth rate is calculated by averaging the rates of all the videos. The numbers appearing to the right of the JSQ legend describe the server bandwidth used for the simulation. For example, JSQ 0.8 used bandwidth of 0.8*(nominal bandwidth). It can be seen from the figure that JSQ reaches low losses only when there is enough statistical gain (i.e. more than seven multiplexed videos) and only when the server bandwidth is high enough (i.e. close to the nominal bandwidth or higher).

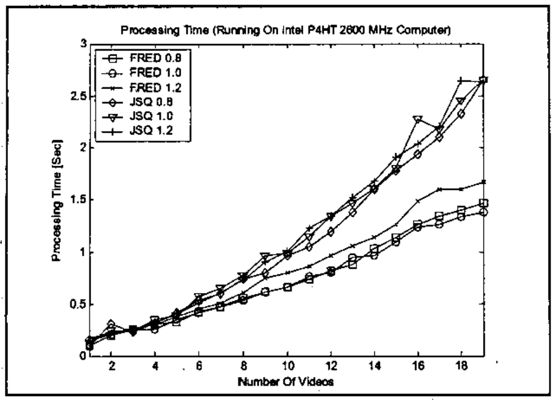

Figure 2. Comparison of processing time of FRED and JSQ algorithms.

Figure 2 depicts the processing time of the two protocols as a function of the statistical multiplexing gain, achieved by multiplexing a growing number of videos. It can be seen that the FRED processing time is always lower than the JSQ processing time. FRED algorithm is completed within 1.5 seconds for multiplexing up to 19 videos. Figure 2 indicates that the complexity of FRED is lower than that of JSQ. FRED algorithm deals with J clients at once, while JSQ algorithm deals with each frame separately. JSQ suits better for real-time decisions, because it does not need to calculate the whole transmission plan to start sending the videos. JSQ need to calculate only part of the transmission plan which enables the clients to display the movie until a specific time. The FRED algorithm lacks this advantage because it has to finish the first phase before it starts processing the second phase. The FRED algorithm can be optimized to reduce the processing time by making the second phase processing time faster than the frames rate. Thus the only processing time of relevance is that of the first phase.

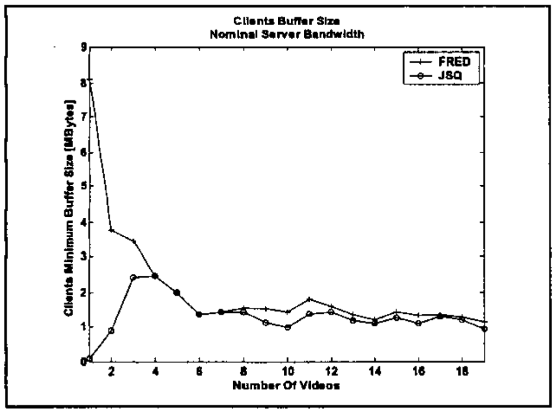

Figure 3. Comparison of clients buffer size of FRED and JSQ algorithms.

Figure 3, shows that JSQ requires lower buffer sizes. We have already seen in Figure 1, that JSQ has high loss probability for the case of less than seven multiplexed videos. This can also be observed in Figure 3. JSQ doesn't smooth very well for less than seven multiplexed videos, therefore it doesn't need a large buffer. When exceeding seven multiplexed videos the statistical multiplexing gain is increased and JSQ reaches low loss probability. Clients' buffer sizes of FRED and JSQ are almost the same with some advantage to the JSQ algorithm. It can be seen that the maximum buffer size requirement of the FRED algorithm occurs for a single video streaming. In this case the clients should have at leave 8MB buffer for lossless receiving. This size of buffer is reasonable for the available technology. Figure 3 indicated the statistical multiplexing gain trend. The linearly descending FRED graph implies that the gain is raised linearly as the number of videos raises (up to 7 videos). The stabilization of FRED graph, which starts from 7 multiplexed videos, implies that the gain reached saturation.

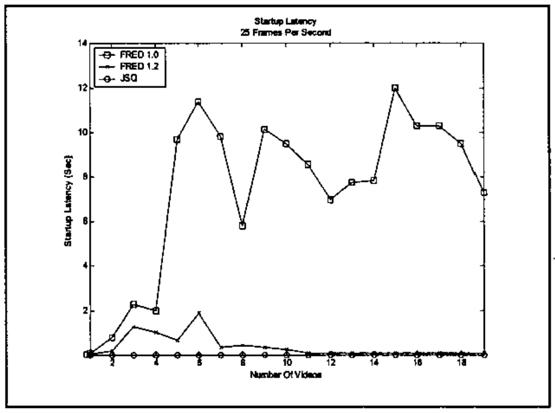

Figure 4. Comparison of startup latency of FRED and JSQ algorithms.

Figure 4 shows that JSQ does not have startup latency at all. This implies that JSQ suits interactive applications. The FRED algorithm does have startup latency. This latency is less than 12 sec for nominal bandwidth and for up to 19 multiplexed videos. FRED startup latency depends on the server bandwidth. Larger server bandwidth should decrease the startup latency (i.e. FRED 1.2 gives only 2 sec latency for the worst case). Lower bandwidth increases the latency. For example, FRED 0.8 has 400 seconds latency for any number of multiplexed videos up to nineteen.

IV. Conclusions

A novel lossless smoothing method, called FRED has been presented. FRED can smooth individual video, but it allows also smoothing multiplexed videos. Smoothing multiplexed streams is more convenient to the network because each video stream statistically smoothes the others. The FRED algorithm can construct a perfect lossless smoothing plan. The costs for this lossless CBR transmission plan are larger startup latency which prevents future losses and minimal client's buffer size requirements, which are tolerable for most applications.

The FRED algorithm is superior to the JSQ when the statistical gain is low or when the server bandwidth is lower than the nominal bandwidth (which is the average bit rate of all the multiplexed videos). In other situations, FRED gives similar results to JSQ. An additional advantage of the proposed algorithm is lower complexity. On the other hand, JSQ has no startup latency, while FRED has latency of several seconds, which is especially a disadvantage for interactive applications. Future work will try to decrease the startup latency of FRED protocol, so it can be adjusted to interactive applications.

The FRED algorithm is very flexible as it guarantees a lossless CBR transmission plan for any server bandwidth. Applications that are indifferent to startup latency should adapt the FRED algorithm as a preferred smoothing algorithm.

Acknowledgment

The authors would like to thank Dr. Oliver Rose from the Department of Distributed Systems, Wuerzburg University for the video streams database. This research was supported by the Israeli Ministry of Science, Grant No. 2105.

REFERENCES

[1] W. Feng and J. Rexford, "Performance evaluation of smoothing algorithms for transmitting prerecorded variable-bit-rate video," IEEE Trans. on Multimedia, pp. 302-313, September 1999.

[2] W. Feng and J. Rexford, "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video," Proc. IEEE INFOCOM, pp. 58-66, April 1997.

[3] W. Feng, "Rate-constrained bandwidth smoothing for the delivery of stored video," Proc. IS&T/SPIE Multimedia Networking and Computing, pp. 316-327, February 1997.

[4] J. Salehi, Z. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: reducing rate variability and end-to-end resource requirements through optimal smoothing," 1996 ACM Sigmetrics Conference, Philadephia, PA, May 1996.

[5] S. Sen, J. Dey. J. Kurose, J. Stankovic, and D. Towsley, "CBR transmission of VBR stored video," SPIE Symposium on Voice Video and Data Communications: Multimedia Networks: Security, Displays, Terminals, Gateways, Dallas, TX, November 1997.

[6] S. Lam, S. Chow, and D. Yau, "A lossless smoothing algorithm for compressed video," IEEE/ACM Transactions on Networking, vol. 4, no. 5, October 1996.

[7] S. Lam, S. Chow, and D. Yau, "An algorithm for lossless smoothing of MPEG video," Proceedings ACM SIGCOMM '94, London, August 1994.

[8] W. Feng and S. Sechrest, "Smoothing and buffering for the delivery of pre-recorded video," Proceedings of ISET/SPIE Multimedia Computing and Networking, pp. 234-244, 1995.

[9] O. Hadar and R. Cohen, "PCRTT enhancement for off-line video smoothing," Special Issue on Adaptive Real Time Multimedia Transmission over Packet Switching Networks in Journal of Real Time Imaging, vol. 7, no. 3, pp. 301-314, June, 2001.

[10] O. Hadar and R. Cohen, "PCRTT enhancement for off-line video smoothing," Proc. SPIE, Multimedia Systems and Applications, vol. 3528, pp. 89-100.

[11] M. Reisslein and K. Ross, "Join-the-shortest-queue prefetching for VBR video on demand," Proceedings of IEEE International Conference on Network Protocols (ICNP), Atlanta, GA, pp. 63-72, October 1997.

[12] O. Hadar, A. Kazir, and R .Stone, "Prefeteched multiplexing of individually smoothed video streams," OptiComm 2001, Optical Networking and Communications Conference, Denver, CO, pp. 19-24, August 2001.

[13] S. Kang and H. Yeom, "Aggregated smoothing: considering all streams simultaneously for transmission of variable-bit-rate encoded video objects," Journal of Communications and Networks, vol. 5, no. 3, pp. 258-265, September 2003.

[14] S. Kang and H. Yeom, "Aggregated smoothing of VBR video streams," Proceedings of the 14th International conference on Information Networking (ICOIN-14), Hsin-Chu, Taiwan, January 2000.

[15] J. McManus and K. Ross, "A dynamic programming methodology for managing prerecorded VBR sources in packet-switched networks," Telecommunications Systems, vol. 9, 1998.

[16] O. Hadar and S. Greenberg, "Video rate smoothing and statistical multiplexing using the e-PCRTT algorithm," SPIE Symposium on Voice Video and Data Communications: Multimedia Systems and Applications II, Boston, MA, September 1999.

[17] Z. Zhang, J. Kurose, J. Salehi, and D. Towsley, "Smoothing, statistical multiplexing and call admission control for stored video," IEEE Journal on Selected Areas in Communications, August 1997.

[18] O. Rose, "Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems," University of Wuerzburg. Institute of Computer Science Research Report Series. Report no. 101, February 1995.

Web address and directory of the used video traces:

http://www3.informatik.uni-wuerzburg.de/~rose/files/MPEGtraces.zip