Real-time Video over Programmable Networked Devices

Tien Pham Van

International Graduate School of Dynamic Intelligent Systems,
University of Paderborn, 33102 Paderborn, Germany
vantien@uni-paderborn.de

Abstract. In this paper, we introduce a novel architecture for programmable network nodes that work with a large number of real-time video streams. We first discuss challenges in transmission of video streams over bandwidth-limited networks, followed by the active approach as an advance for streaming real-time video. In our model, each programmable node makes admission decision for video frames based on evaluating their potential value. Frames "bid" their expected distortion price and the node chooses the best ones first until resource is fully utilized. Analysis of complexity and overhead shows clear benefit of our framework. Simulation experiments demonstrate its consistent outperformance in comparison to lagrangian-based Rate-Distortion Optimized schemes.

1 Introduction

Though various compression techniques have been introduced, networks can still not fully accommodate traffic generated by distributed streaming applications. During congestion, discarding packets is unavoidable, playback process at receiver will consequently suffer additional distortion. Video compression techniques exploit temporal redundancy, thus frames are not equally important with respect to reconstruction process. For example in MPEG4 standard, B frame cannot be decoded without adjacent P frames, likewise P frame must refer to the previous one, all subsequent frames of a GoP (Group of Picture) are considered useless if its I frame is lost.

On enhancing signal quality, several approaches have been proposed. First, modifications of retransmission [1][2] have been made, in which packets are selectively retransmitted in case of loss. This strategy is suitable for non real-time video only due to significant delay and load caused. A proxy selectively caching important video frames may shorten retransmission path [3]. Note however that once congestion occurs at multiple hops, packets may be discarded before reaching the proxy itself. On the track of active network architecture [11], [7] proposes a frame semantics based policy where B frames are dropped first, followed by Ps. This approach improves distortion in a low complexity, but it does not consider difference in size and associated distortion among frames of the same type, thus it is far suboptimal.

Another promising approach is Rate-Distortion Optimized scheme [4][5][6], or RaDiO for short, where transmission policy is formed by minimizing a rate-distortion *lagrangian* function. [6] attempted to bring the model to ordinary active nodes by simplifying frame patterns. Despite variety of proposals have been made, complexity still remains challenging. Heavy complexity obviously hinders the network from accommodating large number of streams. Too much computation forced for optimization may cause excessive processing delay.

Our strategy aims at optimality under IP-based infrastructure with low complexity and overhead. It targets at ordinary active routers and any other type of existing programmable nodes that work concurrently with large number of streams, e.g. proxies/firewalls, WLAN routers, and wireless base stations. In the rest of this paper, we first formulate the problem of optimization in the next section, and then frame-bidding strategy is presented. Next, buffer management is described in section 3. Section 4 discusses implementation aspects and evaluates complexity. Simulation experiments are presented in section 5, showing outperformance with respect to average PSNR and complexity.

2 Frame-bidding Approach

We consider a generic programmable node that needs to forward a set of frames from multiple streams. Decisions on which frame to send, which to drop, and when, will form a *transmission policy*. Each frame is associated with a distortion, which the total distortion at the receiver is reduced if it correctly arrives (hereafter called *distortion reduction*). When overflow occurs, a processor controlling the output interface reconsiders all the frames buffered to find an interim optimal policy. After that, if congestion occurs, the policy is updated with more packets rejected. This tactic lightens effects of prolonged congestion on optimality in a smooth way, and regulates computation load over time. Section 5 shows that a stable playback quality is observed.

2.1 Formulation of the Problem

Let's consider a programmable node that works with *M* video streams that are allocated a buffer capacity *B*. Within a period *T*, accumulated size of accepted frames must always satisfy the following constraint:

$$\sum_{A} S_{mni} \le B + W(T) \tag{1}$$

where *A* is the set of accepted frames, forming a transmission policy π; S_{mni} denotes the total size of frame *i*th of GoP *n* of stream *m* (hereafter referred to as frame *mni*, or *i*); and *W* represents the maximum amount of data that can be dequeued from the buffer within period *T*:

$$W(T) = \int_0^T C(t)dt \tag{2}$$

where $C(t)$ is the bandwidth reserved for all the streams at time t. An optimal policy should maximize total distortion reduction that frames of A account for:

$$\pi^* = \arg\max_{\pi} \left(\sum_{A} D_{mni} \right) \tag{3}$$

where D_{mni} stands for total reduction pertaining to frame *mni*. Due to inter-frame dependency and irregularness of frame arrivals, any online algorithm can give approximate solutions only.

When a frame k arrives and buffer lacks space, some frame(s) must be dropped. Let's denote A_{max} as the set of frames if all remaining frames of the current GoPs are accepted, A_a as the set of currently buffered frames, and J as that of frames to drop. The problem for period T ending when all frames of the current GoPs arrive can be formulated as follows:

$$\frac{B + W(T)}{\sum_{(mni) \in A_{max}} D_{mni} - \sum_{(mnj) \in J} D_{mnj}} \Rightarrow \min \tag{4}$$

$$\sum_{(mni) \in A_a \cup k} S_{mni} - \sum_{(mnj) \in J^a} S_{mnj} \leq B(t) \tag{5}$$

where $J^a \subset J$, containing currently available frames only. Imagine, in exchange for a reduction pertaining to frame *mni*, the node must spend a space equal to its frame size. So *price* per distortion unit can be estimated as (6):

$$p_{mni} = \frac{S_{mni}}{D_{mni}} \tag{6}$$

At best, the buffer and idle bandwidth are fully utilized so that (1) becomes an equality, and the left-side of (4) expresses the average price for all accepted frames. Approximately, the node gains highest total distortion reduction if it rejects the most "expensive" frames, and keeps the buffer full. Because of frame dependency, each frame should be associated with *expected distortion price*:

$$p^e_{mni} = \frac{\sum_{j \in G_i} S_j}{\sum_{j \in G_i} D_j} \tag{7}$$

where set G_i contains i and its dependent frames, excluding those already discarded.

2.2 Bidding Mechanism

Algorithm to locate dropping pattern is illustrated by pseudo code in fig. 1. Whenever the buffer lacks space to accept a new packet of frame k, the interface processor looks back to content of the buffer, calculates expected prices. A list of tag indicating each frame expected price p^e_{mni} together with total size $\sum_{j \in G_i} S_j$ is created. The processor picks out tags with highest price first, records accumulated size, until total size of remaining frames is less than the buffer size, i.e., constraint (5) is satisfied. During the process, if k is hit the loop ends immediately, frame k is rejected, and content of

buffer is kept intact. Once a frame is subjected to drop, its dependent frames must be too.

```
Frame (k) arrives;
free_buffer_space = B - Σ_{A_a} S_mni ;

IF (free_buffer_space >= S_mnk ) admit(k)
ELSE
{
      Calculate p^e for available frames and k;
      Create a list of tag for A_a ∪ k , called tag_list;
      drop_set = Φ /* empty */;
      free_buffer_space = 0;
      buffer_occupance = S_mnk + Σ_{A_a} S_mni ;
      while (buffer_occupance > B)
      {
          drop_set = drop_set ∪ pick(tag_list,
                                     highest_p^e );
          IF (pick(tag_list, highest_p^e ) == frame k)
              BREAK;
          tag_list = tag_list \ highest_p^e ;
          buffer_occupance =
                    total_size(remain_frames);
      }
      IF (k ∈ drop_set)
      {
          drop_set = Φ ;
          Drop frame k ;
      }
      Drop all frames in drop_set;
}
```

Fig. 1. Pseudo code for frame-bidding

3 Buffer Management

In real-time video communication, frames with end-to-end delay exceeding a predefined threshold (typically 500*ms*) are considered unacceptable [9]. Thus, buffer should be properly maintained for higher efficiency. Given that a stream *m* is encoded at frame rate R_m, with GoP length of L_m^{GoP}, the number of frames in buffer at any time should be limited at $\frac{R_m}{2}$ and the number of GoP must satisfy:

$$N_m \leq \left\lceil \frac{R_m}{2 L_m^{GoP}} \right\rceil \tag{8}$$

Specifically, at each frame arrival time, if not both of the above conditions are satisfied, then earliest arriving frames should be deleted to reserve space for the new one. If (8) does not hold, the head GoP in buffer should be completely destroyed.

4 Implementation and Complexity

To ease computation, a logical list is maintained for each stream. Optimization process accesses the list rather than physical packets in buffer. As indicated in fig. 2, packets are first validated at a *GoP Checkpoint*, and then their video header is read and stored in the respective list. When a packet is sent out, the respective list is signaled to update. While queuing, packets may be displaced by new lower price one.

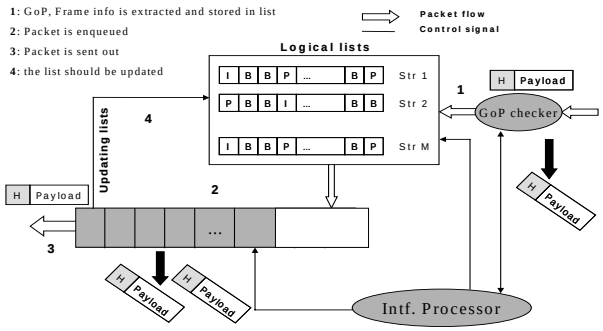

Fig. 2. Logical lists are maintained for streams

What the processor does when overflow occurs is calculating expected price for available frames to form a list of tag. Thus, overall complexity is $O(N)$, where N is the number of frames currently available. Practically, delay exceeding 500*ms* is unacceptable, average frame rate is less than 30 frames/*s*, so the maximum number of buffered frames pertaining to each stream is less than 15. Thus, the worst-case complexity can be expressed as $O(15\times M)$, which is much lighter than that of *lagrangian* approach [6], where the complexity is exponentially proportional to *M*. Like *lagrangian* RaDiOs, correlation with previous GoP can be made to predict the statistics of frames that have not arrived [8]. Video header added to each frame is just to indicate its semantics, total size and distortion. Total number of added bytes is less than 3 per packet.

5 Simulation Experiments

We implement OPNET-based simulation experiments with real-life video streams provided from [10]. Network layout is shown in fig. 3, composed of 10 programmable routers. The four simulated MPEG4 CIF streams are *Akiyo*, *Container*, *Hall*, and *Tempete*, connecting Server1, Sever2, Sever3, and Server4 to fix_rx_1, fix_rx_2, mobile_node_0, and fix_rx_3, respectively. Encoded PSNRs are 38.93dB, 33.96dB, 35.35dB, and 26.04dB, respectively. Nominal bit rate of each streams is approximately 200Kbps. Additionally, a real-life trace-based FTP data flow is used to cause further congestion. Each original MPEG4 episode has 300 frames, but is repeated to

have 6000 frames in total. Simulation experiments were conducted in both our proposed strategy and *lagrangian* RaDiO.

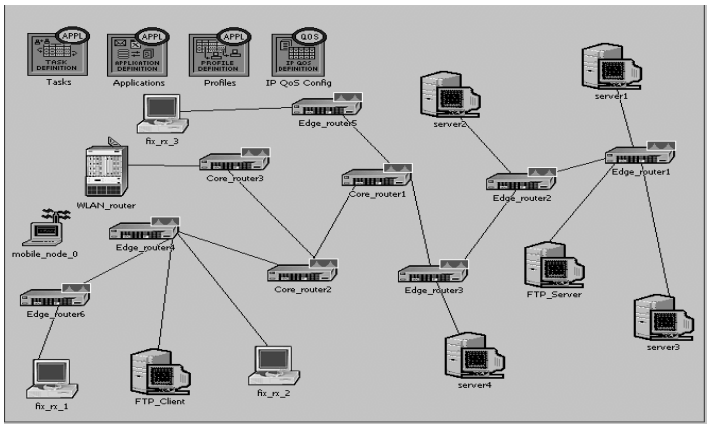

Fig. 3. Network layout

5.1 Distortion

A sample of reconstructed video is shown in fig. 4. One can easily notice better perception quality in our strategy. PSNR of *Akiyo* is illustrated in fig. 5 as a sample, showing stable quality in our strategy.

a) fr-bidding b) lagrangian RaDiO

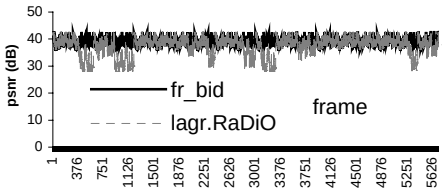

Fig. 4. A sample frame in two strategies

Fig. 5. A stable PSNR is observed in our case

First, we fixed buffer size at 40Kbytes and changed capacity of links so that the network experiences from severe to mild congestion. As indicated in fig. 6, PSNR in our strategy is consistently improved, up to 3.89dB. If all routers are passive, quality of reconstruction video is almost unacceptable. When congestion is not too severe, as buffer size increases, the improvement is clearly noticed (fig. 7). The reason is that our strategy admits more important packets whereas the buffer *lagrangian* RaDiO may unnecessarily reject.

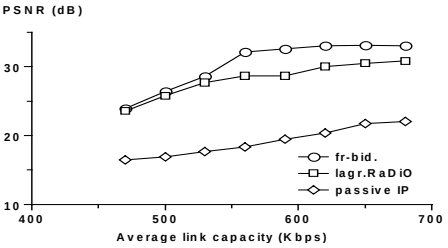

Fig. 6. PSNR vs. average link capacity

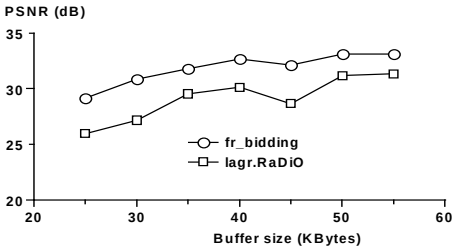

Fig. 7. PSNR vs. buffer size

5.2 Delay

We tuned both buffer size *B* and average link capacity *C*, and collected delay statistics. Though queues tend to be longer in our strategy, no major difference between the two cases is observed (only several *ms*), as indicated in fig. 8. Packets in *lagrangian* RaDiO are dropped as soon as buffer fullness is greater than a threshold ($B_{\min}$). Two different thresholds are separately simulated, as $B_{\min}$ is reduced from 75% of total reserved capacity *B* to 66%, delay slightly decreases.

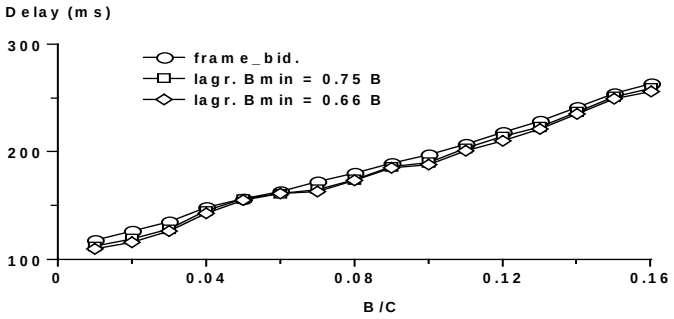

Fig. 8. Delay vs. ratio of buffer size and link capacity

5.3 Run-time and Complexity

OPNET is discrete event-driven simulator, so run-time duration does reflect algorithmic complexity. In our approach, run-time length is approximately 20% shorter than *lagrangian*-based RaDiO. In a large network scenario with 8 video streams, simulations of *lagrangian* RaDiO hang halfway with 8 nested loops at each router to scan all possible dropping patterns [6]. In contrast, the simulations in our framework ran smoothly.

6 Conclusion

Toward enhancing transmission of real-time video, we have proposed a frame-bidding approach, taking complexity and overhead into account. Simulation experiments clearly demonstrate outperformance of our framework, regarding signal quality and computation complexity. The approach is especially suitable for real-time and interactive multimedia communication since neither retransmission nor acknowledgement is needed. Remarkably, complexity in our model is linearly proportional to number of streams, which is feasible for nodes that work with large number of streams.

In the next effort, we will consider whether cooperation between active routers further enhances end-to-end transmission performance. We also foresee the ability of integrating our model into path-diversity scenarios to extend aggregated bandwidth. At present, we implement the framework on a testbed composed of several Linux PC-based routers, with WLAN connections to end-users.

References

1. G. B. Akar, N. Akar, E. Gurses, “Selective Frame Discarding for Video Streaming in TCP/IP Networks,” Packet Video, 2003.
2. Argyriou, A. Madisetti, V., “Streaming H.264/AVC video over the Internet,” IEEE Consumer Communications and Networking Conference, 2004.
3. I. Bouazizi, "Size-Distortion Optimized Proxy Caching for Robust Transmission of MPEG-4 Video," In LNCS 2899, Proceedings International Workshop on Multimedia Interactive Protocols and Systems, November 18-21, 2003.
4. E. Masala, H. Yang, K. Rose and J. C. De Martin, “Rate-Distortion Optimized Slicing, Packetization and Coding for Error Resilient Video Transmission,” Proceedings of IEEE Data Compression Conference, 2004.
5. P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media", IEEE Trans. Multimedia, 2001, submitted.
6. W. Tu, W. Kellerer, and E. Steinbach, "Rate-Distortion Optimized Video Frame Dropping on Active Network Nodes," Packet Video Workshop, 2004.
7. G. Ravindra, N. Balakrishnan, K. R. Ramakrishnan, “Active Router Approach for Selective Packet Discard of Streamed MPEG Video under Low Bandwidth Conditions,” Proc. ICME 2000, 2000.
8. Z. He and S. K. Mitra, "A Unified Rate-Distortion Analysis Framework for Transform Coding," IEEE Transactions on Circuits and Systems for Video Technology, VOL. 11, NO. 12, December 2001.
9. X. Meng, H. Yang and S. Lu, "Application-oriented Multimedia Scheduling over Lossy Wireless Networks," IEEE ICCCN 2002, October 2002.
10. J. Klaue, “YUV CIF video sequences”, http://www.tkn.tu-berlin.de/research/evalvid/
11. Y. Bai and M. Robert Ito, “QoS Control for Video and Audio Communication in Conventional and Active Networks: Approaches and Comparison,” IEEE Communications Surveys & Tutorials, Vol.6, No.1, 2004.