CSE 534 Project Report

Re-implementation of Packet Scheduling Algorithm for Wireless Video Streaming

Yifang Cao (109330080)
GitHub:

1.Introduction

The project is intended to re-implement the packet scheduling algorithms for wireless video streaming as mentioned in the paper of Sang Kang and Avideh Zakhor[1]. Traditionally, the most commonly used scheduling mothed was earliest deadline first scheduling(EDF), which sends video frame packets sequentially in order of the deadline. Under good network connections, EDF is efficient in terms of minimizing required buffer size at the receiver side because the packets always arrive in order and the receiver does not need to have a large buffer. However, the performance of EDF is limited under error prone connections, for it does not handle importance of data in encoded video stream.

In section 2, the video encoding technique and the limitation of EDF are explained in detail. The paper introduces a new frame based packet scheduling algorithms(FBS) for MPEG-4 encoded video frames to mitigate the issue of EDF. The new algorithm can preserve video streaming quality by reordering transmission schedule based on the importance of each video frame. Section 3 introduces FBS algorithm and explains how the network environment is set up and FBS is implemented in this project. In Section 4, the performance of EDF and FBS are evaluated and compared with various loss metrics. The evaluation has shown that in general FBS is able to transmit more important frames than EDF. **With a low loss rate, FBS behaves similarly to EDF. However, with a very high loss rate, FBS only sends the most important frames and tends to ignore other frames. Also, because FBS adds the retransmission mechanism, there are a lot more retransmissions with highly lossy connections than with good connections.**

2.Conventional Video Streaming

2.1 Video Encoding

The purpose of live streaming is to deliver digital content in real-time through Internet connections. The main challenge of streaming is that if the Internet connection does not have enough bandwidth, the receiver may experience intermittent stops and frame jumps. To make sure that the large digital videos can transmit quickly enough, videos are usually compressed at the sender side. In the most commonly used compression standard, H.264/MPEG-4 AVC, the video frames can be compressed into three frame types, I frame, P frames and B frames. The project will only focus on I frames and P frames. I frame are standalone video pictures, and servers as the base frame for its following frames. P frames usually only contains the motion meta data, and requires previous frames to decompress.

As shown in Figure 1, MPEG-4 compression can partition a video into many groups of pictures(GOP). Each GOP consists of one leading I frame and several following P frames. Each

video frame also consists of several network packets. P frames can be decompressed through the data from previous frames, while I frames are least compressed frame and holds which carry the base information for the following P frames and cannot be decompressed through other frames. Therefore, if an I frame is unable to make it to the receiver side or highly corrupted at the time of display, the video quality will be adversely affected a lot. Also, the frames in the front of each GOP are more important than the latter frames, for the losses of frames in the front can break the complete decompression chain earlier. A transmission scheduling algorithm is needed to determine which video frame should be sent at a certain point of time.

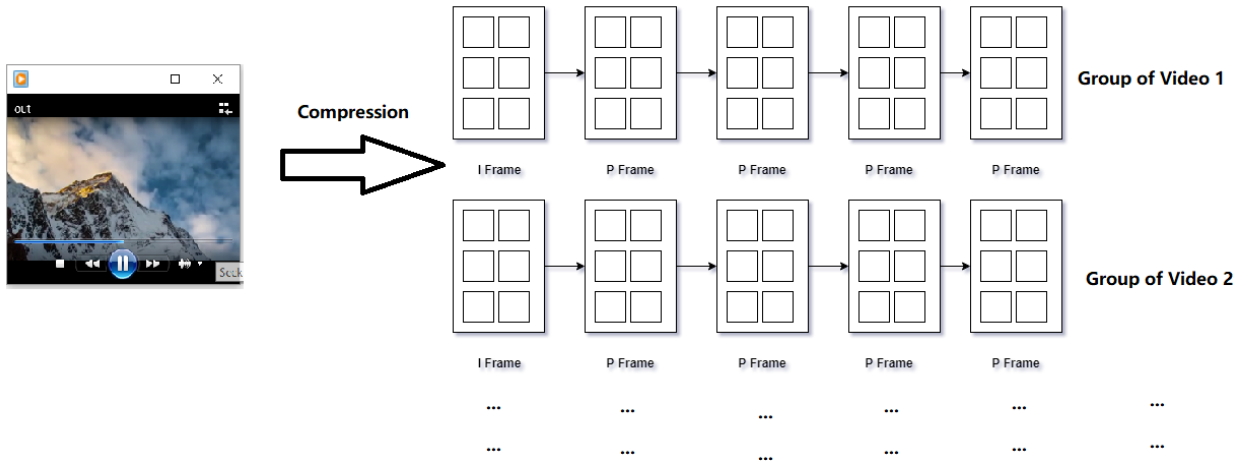

Figure 1 – Video Compression

2.2 Limitations of Earliest Deadline Frist Scheduling

The earliest deadline first is a commonly used simple algorithm to schedule frame transmission. This algorithm determines the sending order based on the deadline of each video frames, which is the display time at the receiver side. Because the video always displays in sequential order, EDF in fact simply sends each video frame in order. The advantage of this algorithm is that it can minimize the dwelling time of video packets in the receiver's buffer because the video packets are highly likely to arrive in order. However, deadline-only ordering criteria and sequential transmission do not consider the importance of each video frame. As discussed above, the losses of earlier frames in a GOP can highly affect the video quality. Therefore, besides the deadline criteria, a better scheduling algorithm should also give high priority to the earlier frames in a GOP.

3. Frame Based Scheduling Algorithm

3.1 Algorithm Explanation

FBS is a UDP based algorithm with additional acknowledgement mechanism. If a frame is received at the receiver side, the receiver sends acknowledgement back to the sender. The algorithm is applied at the application layer. In the preprocess phase, FBS firstly assigns each frame a frame type α by the first equation below based on the position in a GOP, where *i* is the frame number and *M* is the size of the GOP. Basically, the more important frame will be assigned with a smaller frame type number, and the less important frame will be assigned with a larger number.

$$\alpha_i = i \mod M$$

$$d(VP_{i,j}) = \frac{\beta\alpha_i}{M-1}, \text{[sec]}$$

Then, the deadline threshold *d* is calculated for each video frame by the second equation above. The β is a tunable variable called importance coefficient. This variable can control the extent of the importance criteria, and its value is usually tuned by the receiver's buffer size or statistics of burst error. The important frames will have a smaller deadline threshold. At the time of each transmission, FBS uses two-pass algorithm to determine which frame should be sent:

In the first pass, FBS chooses the first unsent frame as the candidate that satisfies the following constraint:

$$D(VP_{i,j}) > d(VP_{i,j}) + \Delta,$$

D is the difference between the current time and display deadline of the candidate frame. The current time can be easily calculated by $t = n/f$, where *n* is the current frame number at display and *f* is the frame rate of the video. Similarly, the deadline of the candidate frame can be calculated by $t = i/f$, where *i* is the candidate frame number. The variable Δ denotes the time latency between the time the frame is sent at the sender and the time the frame is received at the receiver side. Its value can be estimated by RTT. This constraint will ensure that the candidate chosen should be able to make it to the receiver at the time of display.

In the second pass, FBS searches backwards to see if there is any frame with the same or smaller deadline threshold *d*, which has been sent out but not acknowledged. The earliest satisfied frame will be eventually sent. In this case, the potentially lost frame gets a second chance to be transmitted if the deadline threshold permits.

3.2 Environment Setup and Implementation

The wireless network environment simulation was done by MiniNet. Figure 2 demonstrates the topology of the network setup. The network consists of 2 hosts (1 sender and 1 receiver), and 1 switch in the middle. The link on the left is a normal link, and the link on the right is a traffic control link with a loss rate of 5% to simulate the error prone wireless connection.

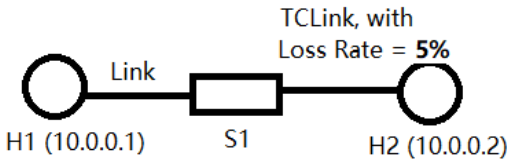

Figure 2 – MiniNet Topology

The application was implemented with multi-thread Python socket programming plus OpenCV. OpenCV is integrated with FFmpeg, this may allow more flexibilities for video processing. Figure 3 demonstrates the structure of the application.

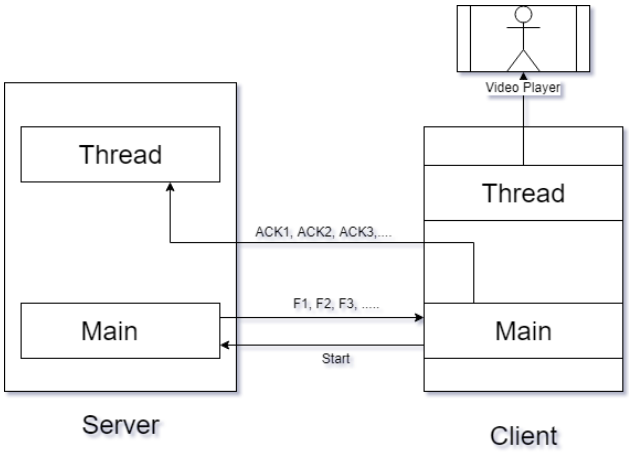

Figure 3 – Multi-thread Socket Programming Implementation

Sender side: The server has two threads. The main thread is used to send out frame packets based on the scheduling algorithm using UDP, and mark the sent frames. Both EDF and FBS were implemented here. The secondary thread listens to another port and is used to listen to the acknowledgement sent back from the receiver, and mark the acknowledged frames.

The sender: Initially, the main thread sends out a start signal to the server to start transmitting video frames. The signal can tell server which scheduling algorithm to use. Then, the main thread reconstructs the received video frames, sends back acknowledgements for the completely received frames, and stores the frames into the receive buffer. Partially received frames will be dropped and no acknowledgements are sent back for these frames. After the first frame is received, the secondary thread simultaneously displays the video frames stored in the receive buffer at a frame rate of 25 frames/sec. The video is displayed using OpenCV's imshow().

4. Performance Evaluation

To evaluation the performance of EDF and FBS, a video clip of 20 seconds long was used to perform the simulation: 'Mountain Climbing', 240x320, 510 frames in total, f = 25 frames/sec. **The evaluations were performed with a loss rate of 5% in Section 4.1, 4.2, 4.3 and 4.4. The evaluation of FBS with different loss rates is in Section 4.5.**

4.1 Frame Loss Distribution

Frame loss distribution is not a fine-grained measurement for the performance of the two scheduling algorithms. However, it can give high level intuitions about how these two algorithms work. Because the important frames in FBS have high priorities and have more changes to be retransmitted, we expect that FBS has different packet ordering from EDF and there should be less losses on those important frames. Figure 4 shows the overall frame loss distributions of both EDF and FBS. Each vertical bar represents a frame. The bars with value of 1 represent those successfully received video frames, and the bars with value of 5 represent the lost frames, which were the frames that were not fully received at the time of display. The lost bars for EDF are more relatively random. The lost bars for FBS are more evenly spaced out, indicating there is a loss pattern there.

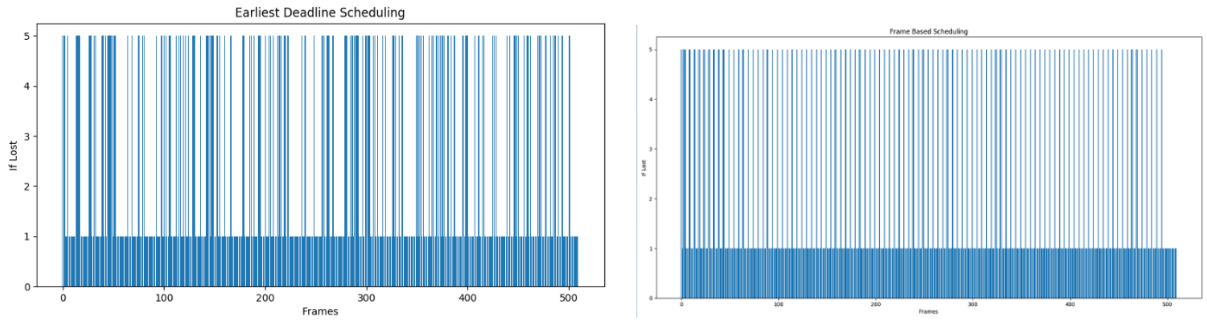

Figure 4 – Frame Loss Distribution.
The left graph is for EDF, and the right graph is for FBS.

Figure 5 shows the first 50 frames' loss distribution of the two algorithms. The red vertical line indicates the starting frame of each GOP. With a closer examination on the loss distribution of only the first 50 frames, it has clearly showed that FBS successfully avoids the earlier frame losses in each GOP.

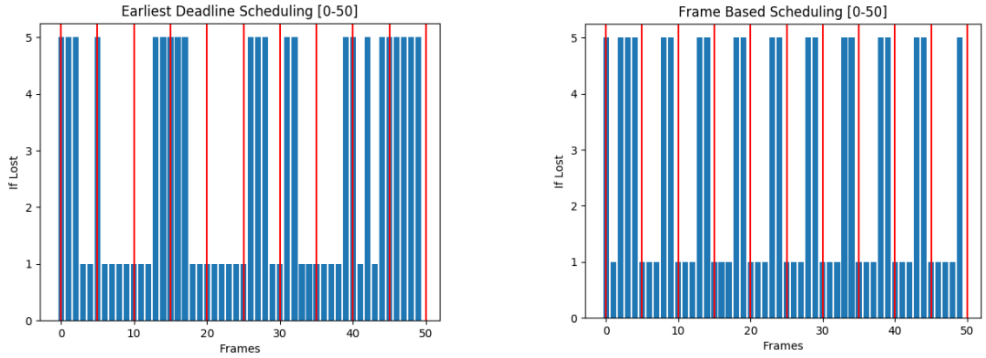

Figure 5 – Frame Loss Distribution (First 50 Frames only).
The left graph is for EDF, and the right graph is for FBS.

4.2 Loss Number

As discussed in Section 2, I frame are the base picture to the entire GOP. Successfully transmitting I frames are extremely critical to ensure the video quality. The following graphs show how the two algorithms preserve I frames. For EDS algorithm, there are 170 losses in total, out of which there are 37 losses of I frames, roughly 21.7% of the total loss. For FBS, there are 121 losses in total, but only 1 loss of I frames. FBS performs much better on preserving I frames.

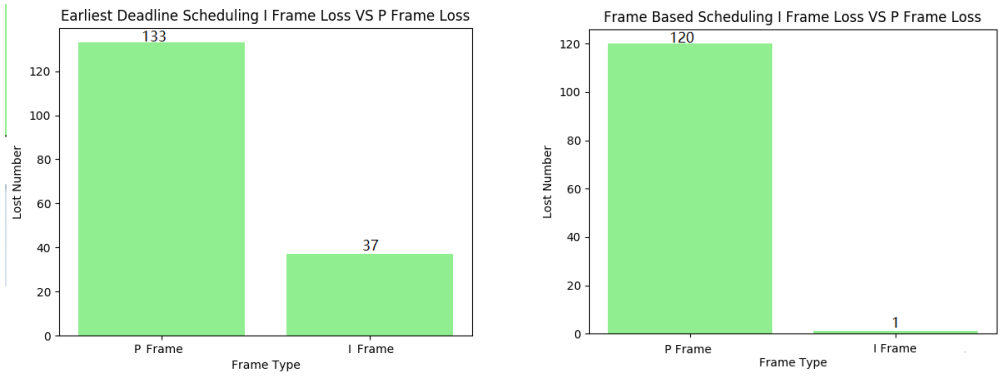

Figure 6 – Loss Number of two Algorithms.

The reason that the total losses of EDF are higher than that of FBS is that FBS has retransmission mechanism so that the lost frames have second chance to be transmitted, leading to less losses. Hence, for the total of 510 video frames, EDF has a 33.3% overall loss rate, while FBS only has a 23.7% overall loss rate.

4.3 Type Loss Ratio

In the preprocess phase, FBS assigns a frame type to each video frame. Lower frame types are for the more important frames. Figure 7 shows the loss ratio for each frame type. We can tell that for EDF the loss ratio on each frame type is all evenly around 20%; there is no loss bias for EDF. The loss ratio of FBS is biased. FBS loses fewer important frames, and has high loss ratio only on those less important frames. This implies that FBS is able to preserve better video qualities by successfully transmitting more important video frames.

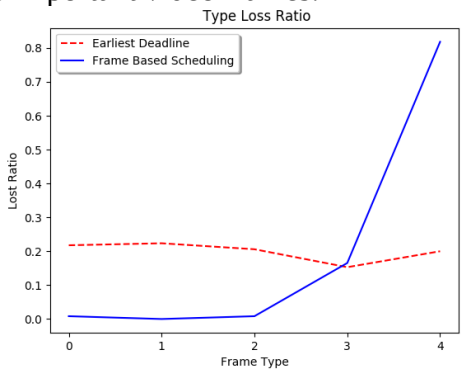

Figure 7 – Type Loss Ratio. The red dashed line is for EDF and the blue solid line is for FBS.

4.4 Visual Disparity

With EDF, the video displayed at the receiver side has more frame jumps and jumps are random. With FBS, the video displays more smoothly at the server side and has fewer frame jumps. And the frame jumps are more arithmetic in compared to EDF.

4.5 Performance with Various Loss Rates

Besides with 5% loss rate, the performance of FBS was also evaluated with 10% loss rate and with 20% loss rate. Figure 8 shows the sending behavior of FBS with different loss rates. The figure indicates that FBS sends out more important frames with high loss rate. With low loss rate the frames of each type are sent out much more evenly, which is similar to DEF.

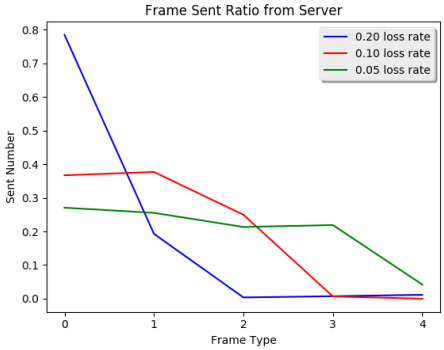

Figure 8 – Frame Sent Ratio from Server. The blue line is for 20% loss rate, the red line is for 10% loss rate and the green line is for 5% loss.

Another statistic worth noticing is that the number of frames sent out from server with 20% is extremely high; it sends out 8453 frames out of the sender side while there are only 1046 with 10% loss rate and 1022 with 5% loss rate. This implies that re-transmission number increases significantly when the loss rate gets higher. FBS tries very hard to make sure that important frames can arrive at the sender side, and almost ignores those less important frames.

5.Conclusion

The project provides an implementation of the frame based scheduling algorithm based on both frame deadline and importance criteria. The reason why the performance of the conventional earliest deadline first algorithm is limited under error prone connections is also elaborated, which is that the earliest deadline first algorithm has no re-transmission mechanism and fails to ensure the successful transmission of the earlier frames in each group of pictures. The frame based scheduling algorithm addresses this issue by a two-pass searching algorithm before each transmission, which give high priorities and more re-transmission changes to the important frames in each group of pictures. The evaluation and comparison between the earliest deadline first algorithm and the frame based scheduling algorithms validate that the frame based scheduling algorithm outperforms with lossy connections. With low loss rate, the two algorithms essentially behave similarly. With high loss rate, the frame based scheduling is able to make more number of retransmissions on the important frames, and ignore less important video frames; thus, the frame based scheduling better preserves the streaming video quality with lossy connections.

6.References

[1] Kang, Sang H., and Avideh Zakhor. "Packet scheduling algorithm for wireless video streaming." *International Packet Video Workshop*. Vol. 2002. 2002.