

Architecture for Multi-Client Multi-Channel Compressed Video Streaming

R.G.J. Wijnhoven, M.C. Jacobs, P.H.N. de With, E. Jaspers
Eindhoven University of Technology / Bosch Security Systems B.V.,
Eindhoven, The Netherlands
E-mail : R.G.J.Wijnhoven@student.tue.nl

Abstract — In this paper we describe an architecture for a multi-client, multi-channel video streaming server targeted at the security market. For each connected client, the best selection of available compressed video streams per channel is made, optimizing the perceived quality of the video channels requested by the client. We propose techniques to scale the bitrate per video stream and introduce a scheduling scheme to select the best video streams for a given set of channels and bandwidth constraints.

Keywords — Architecture; streaming video; multi-client; multi-channel; scheduling; bitrate scaling; restricted shortest path problem

I. INTRODUCTION

In the video security market, digital video recorders (DVR) have been available for some time. DVRs store video from multiple channels (cameras) and offer remote display of recorded and live video over a network connection. Multiple clients can connect to the system and view video simultaneously (see Figure I).

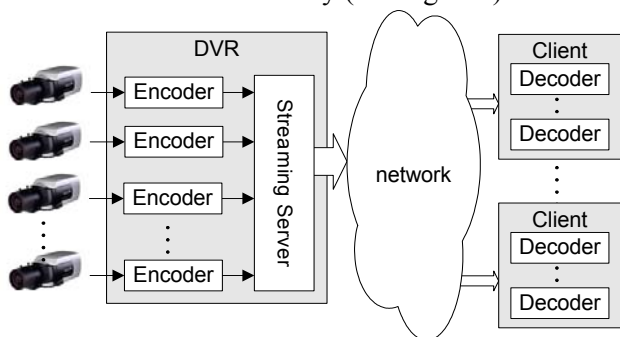


Figure I: Use case.

DVRs often use still-image-based compression techniques like wavelet- or JPEG-compression. To increase video quality and storage times, there is a shift to compression techniques that exploit the temporal correlation of video. These techniques are called inter-coded compression and are often based on one of the

MPEG standards [4]. MPEG defines a group of pictures (GOP) as one independently decodable frame (I-frame) followed by a number of inter-predicted frames (P-frames). Such a predicted frame is encoded using the difference between the current frame and the motion-compensated previous frame. Consequently, higher compression factors can be achieved. However, due to the dependencies between encoded frames (P-frames), all preceding data in a GOP needs to be transmitted and decoded in order to decode a certain frame. Skipping frames without considering the coding scheme will result in errors in the decoded video. Thus, it is hard to adapt to changing network conditions compared to simply dropping frames when using still-image-based compression. Thus, scalability of inter-coded video is less straightforward.

MPEG defines scalable profiles like the Fine Granularity Scalability profile (FGS), that include spatial, SNR and/or temporal scalability. We do not consider these profiles, because they are too computationally intensive.

Most systems currently on the market that transmit inter-coded video are single-client based, or broadcast a single video channel to multiple-clients. Furthermore, they don't offer scalability in bitrate: only a limited number of (pre-)compressed streams is available. Examples are streaming video servers on the Internet: watching movie trailers (on-demand) or broadcasts of concerts (live).

A proposal for multi-channel video encoding for broadcast over a network with fixed bandwidth can be found in [1]. A complexity measure per video channel is proposed, to partition the total bandwidth over the total number of channel encoders. The video encoders are dynamically controlled to obtain the target bitrates. This solution cannot be used, since the bandwidth is not fixed and the encoders cannot be dynamically controlled (see Section III).

II. DOCUMENT LAYOUT

Chapter III describes the problem. The proposed architecture and its components are discussed in Section IV. The scheduling algorithm, used to select the best streams for a given set of channels, is explained in Section V. This paper ends with conclusions and future work.

III. PROBLEM DEFINITION

In the considered system, the following constraints are identified:

- Bandwidth limitation by the server (e.g. set by the system administrator) or by the network constraints.
- Multiple clients can be connected, each viewing multiple channels.
- Because one source coder can have multiple consumers (multiple connected clients and storage on disc) the parameters of the source coders cannot be changed dynamically. This makes the system independent of the used inter-coded based source coder. Also, the target bitrate (CBR/VBR) per video stream is specified.
- There is limited computing power available.
- Video quality requirements for surveillance purposes are different from consumer entertainment; lower resolutions and frame rates are accepted, also to increase storage times.

The objective of the streaming server is to obtain the best perceived quality for connected clients, given the mentioned constraints.

IV. ARCHITECTURE

Figure II shows the overall architecture of the video server. It is divided in four main components: the source reader, the bitrate scaler, the dispatcher and the controller. Each will be described separately.

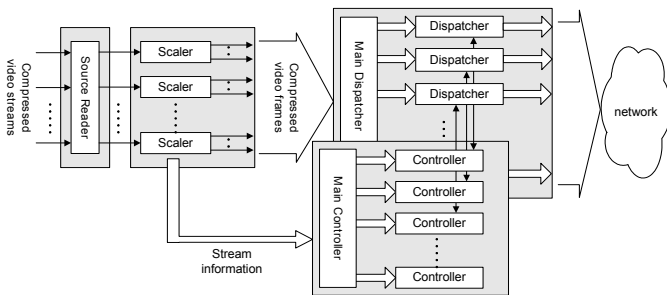


Figure II: Architecture of the multi-client, multi-channel video streaming server.

The actual transmission of data over the network and the measurement of network bandwidth is beyond the scope of this paper. Information on this subject can be found in [2] and [3].

A. Source Reader Component

The source reader receives compressed video frames (for all channels) from the source and transmits them at the display rate. Each video channel can have multiple source streams. Inputs of this component can be real-time video encoders, but also disc readers (in case of stored video).

The video streams are sent to the bitrate scalers. These scalers create streams with lower bitrates (see Section B).

B. Bitrate Scaler Component

This component contains a scaler for each source stream. Each scaler is able to scale a source stream to a limited number of output streams, each having a different bitrate. The bitrate of the output streams is calculated and together with other information about each stream sent to the controller component.

Let us consider methods for decreasing the bitrate of the video streams:

- Transcoding: video streams are partially decoded and consecutively encoded at different rates. Although this solution offers graceful degradation of video quality for a fine granular scalability in bitrate, the required system resources could exceed the available budget.
- Using multiple compressed source streams per video channel, each with a different frame rate. These streams are created before entering the source reader. They require extra processing costs in the encoders, although streams at low frame rates are relatively computationally inexpensive.
- Frame dropping: decode only some of the P-frames of each GOP, and display only the ones at regular intervals. See the example in Figure III, where the frame rate is reduced with a factor of three, while dropping only two out of six frames. Fractions of the original frame rate can be obtained, yielding a relatively small bitrate reduction. The method is computationally expensive for the decoding at the client, in comparison to decoding a regular stream at the same frame rate. The simplest case of frame dropping is the case where all P-frames are dropped, and only I-frames are transmitted. If even more reduction in bitrate is needed, also I-frames can be dropped.

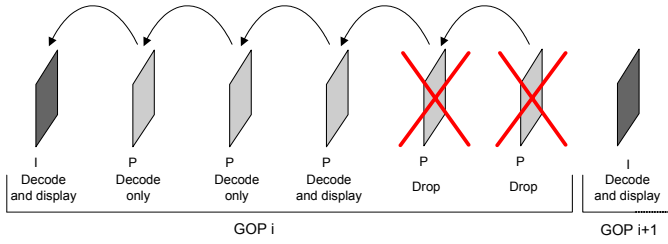


Figure III: Frame dropping

For our implementation, we choose not to apply transcoding because of processing constraints. Per channel we use a limited number of predefined streams, having different bitrates, and implement frame dropping. The number of encoders and the choice of encoder parameter settings for the streams is very important. However, this is out of the scope of this paper.

C. Dispatcher Component

The dispatcher component contains one main dispatcher, and a client dispatcher for each connected client. The main dispatcher receives compressed frames from the scaler component. Each frame is part of a video stream and is sent to each client dispatcher that requires the stream. The selection of the streams is calculated and updated by the corresponding client controller. Subsequently, each client dispatcher concatenates all received frames and forwards them to the network buffer for transmission.

D. Controller Component

As in the dispatcher component, one main controller is available, and a client controller for each connected client. The main controller receives information about the streams from the bitrate scaler component. When the main controller receives information regarding a change in available bandwidth, rescheduling is started. The main controller divides the available total bandwidth over the client controllers, which independently execute the scheduling algorithm (see Section V).

V. SCHEDULING ALGORITHM

Inputs for the scheduling algorithm are the available network bandwidth for the client, the channels that are requested by the client and the available streams per channel. Properties of each stream are the bitrate, frame rate, resolution, GOP-length, time to next I-frame, I-/P-frame size ratio and the currently selected stream for this channel. For each channel, a stream needs to be selected, giving the overall optimal quality over all channels, while the total bitrate does not exceed the bandwidth budget.

To solve this problem, we construct a tree, where each

level in the tree represents a channel. Each edge represents a stream and has two values: a feasibility value (bitrate) and an optimality value (quality value). Each child node of an edge will be the parent node for the next channel. Each leaf represents a stream combination for the used channels. An example of a tree with two channels, each having three different streams, is shown in Figure IV.

The algorithm needs to select the optimal solution from the set of leaves. Each leaf represents a combination of streams (a unique path from the root to the leaf). The sum of the bitrates over the path needs to be less than the bandwidth budget. From this subset we select the path with the highest quality value.

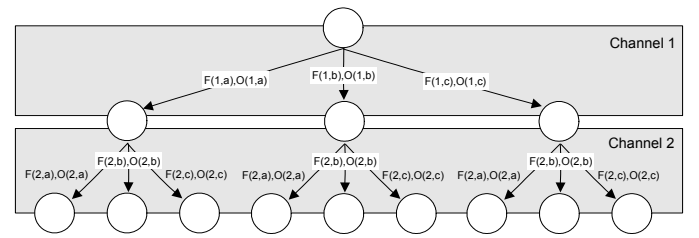


Figure IV: Tree for scheduling, with $F(c,s)$ the feasibility value of channel c and stream s and $O(c,s)$ the optimality value.

This problem is known as the Restricted Shortest Path (RSP) problem and is NP-complete. Extensive research in this field has been done in the area of QoS network traffic routing. An overview of (heuristic) selection algorithms is given in [5].

A. Implementation

The streams are stored in the tree, sorted to bitrate. We use a depth-first search to calculate all the feasible paths, starting with the lowest bitrate streams. When the sum of bitrates of streams on a path fits the available bandwidth, the path is stored. If the bandwidth budget is exceeded, we stop the algorithm because the streams are sorted to bitrate.

From the remaining set of feasible paths, we select the path with the highest quality value.

B. Stream Quality Value

We propose to use a combination of stream properties to calculate the quality value. The following properties are used:

- Frame rate: higher frame rates correspond to a higher perceived quality.
- Bitrate: when two streams have the same frame rate, the stream with lowest bitrate will have a higher quality value (assuming an equal perceived quality of the decoded video).

- I-/P-frame ratio: in case of motion, P-frames have a relatively larger size. For video with motion, higher frame rates are preferred, compared to streams with little motion.
- GOP-length: closely related to time to next I-frame. If GOP-length is high, time to next I-frame will also be high on average.
- Time to next I-frame: if the bandwidth budget decreases, we want to send less bits to the network, so a higher time value is desired. On the other hand, for very low frame rates, a lower time value is preferred.
- Currently selected stream: it is preferred not to switch streams for a channel too often.
- Resolution: higher resolution leads to higher perceived quality.

VI. CONCLUSIONS

We proposed an architecture for an embedded streaming server that is independent of the used source coders and type of network connected. The number of input sources and bitrate scaling methods is variable, as is the number of connected clients. A scheduling algorithm selects the optimal combination of streams with graceful degradation, depending on the used bitrate scaling methods.

First results of a PC-based software implementation look promising and provide a good base for further research.

VII. FUTURE WORK

The performance of the scheduling algorithm is heavily dependent on the choice of the quality value calculation. Therefore, more work needs to be done, to find a good choice for combining the various properties of a stream into one quality value.

Although each client scheduling algorithm will return the optimal stream selection for the given bandwidth budget, the overall solution may not be optimal, since the sum of the client bitrates may be much less than the overall bandwidth budget in the main controller. Therefore, the algorithm has to be extended to find a globally optimal solution.

REFERENCES

- [1] G.J. Keesman, *Multi-Program Video Data Compression*, Ph.D. Thesis, TU-Delft, October 10, 1995
- [2] N.G. Feamster, *Adaptive Delivery of Real-Time streaming Video*, M. Eng. Thesis, Massachusetts Institute of Technology, May 2001
- [3] W. ten Kate, *Internet Streaming: Transporting Continuous Media*, in Proc. Embedded Video Streaming Technology (MPEG-4) and the Internet, IEEE, EESI, Eindhoven, The Netherlands, Dec. 20, 2003, Page(s) 119-144
- [4] Moving Picture Experts Group (MPEG) Homepage, <http://mpeg.telecomitalia.com/>
- [5] F. Kuipers, P. van Mieghem, *An overview of constrained-based path selection algorithm for QoS routing*, Communications Magazine, IEEE , Volume: 40, Issue: 12 , Dec. 2002, Page(s): 50 –55