A Layered Video Transcoder for Videoconference Applications

Gianluca Iannaccone, Luigi Rizzo
Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica e Telecomunicazioni
Università degli studi di Pisa, Via Diotisalvi 2, I-56126 Pisa, Italy

Abstract— **Recent research works have proposed the use of layered schemes to distribute multimedia streams to multiple heterogeneous receivers, stimulating research on (and making use of) algorithms for layered video *compression*.**

In this paper we follow a slightly different approach, presenting a low complexity layered video *transcoder* (`lvt`) suitable for network-friendly multicast videoconference transmission. The transcoder approach permits the exploitation of layered distribution schemes with existing video tools or even with recordings, without necessarily having to act at the data source, and thus providing an easy path to the deployment of such services.

We have implemented the proposed transcoder as a standalone application that cooperates with a widely used videoconference application like VIC. `lvt` intercepts VIC output data and generates a four-layer video stream with the same aggregate data rate, suitable for use with multicast congestion control algorithms such as the Receiver-driven Layered Congestion control (*RLC*). `lvt` performs well even at low bit rates, effectively adapting the source rate to the desired rate.

In the paper we present the architecture of `lvt` and provide an accurate analysis of performance of the proposed layered video transcoder using parameters such as image quality, frame-rate, robustness and data redundancy.

Keywords— **Multicast, Congestion control, Layered encoding, Videoconference**

I. INTRODUCTION

The design of an efficient protocol for communications in a multicast environment has to cope with the main problem of receivers heterogeneity. Internet success and its world-wide diffusion has been possible mainly because of the capability of operate on a variety of computer systems, largely different in terms of hardware architecture and bandwidth availability.

In a multicast environment, this peculiarity makes it difficult, almost impossible indeed, to use a single fixed rate to satisfy all users interested in a given transmission. If the source of data chooses a transmission rate of 256 Kbps, for example, all users with a lower available bandwidth will experience a very high loss rate, given by the foreseeable congestion of nearby routers, while users with a higher bandwidth availability might complain for the poor source quality with their system resources underutilized .

An approach to solve this problem, that is gaining an increasing consensus among the research community, is the use of layered data streams [3], [4], [6], [12], [14]. In this model, the source distributes multiple levels of quality of the same signal simultaneously across multiple network channels. If data are organized in a layered way, receivers may adapt their reception rate to bandwidth availability choosing which layer, or how many layers, to receive. This technique is especially suitable for real time multimedia transmissions, but also for bulk data transfers [9], [14], where quality of signal has to be intended as transfer speed.

In order to realize this model, a specific multicast protocol has to be introduced. This protocol must permit the selective forwarding of subflows along all the links and has to make receivers able to evaluate bandwidth availability and to act coordinately.

The former aspect, i.e. selective forwarding of subflows, is achieved using the Internet Group Management Protocol (IGMP) [1]. The source of data transmits layers across multiple multicast groups, while receivers join and leave these multicast groups with IGMP messaging, choosing in this way which layers will be received.

To grant the effectiveness of this model, some form of co-ordination between receivers must be provided. Indeed, if two receivers in the same subnetwork choose different layers (or *subscription level*), either the receiver with the lower subscription level does not fully exploit available bandwidth, or the receiver with the higher subscription level is causing congestion over the network.

A feasible solution to address these aspects of multicast transmissions is the Receiver-driven Layered Congestion control algorithm (*RLC*) [14]. RLC, which is inspired and improves a previous work scheme called RLM [6], is designed to support one-to-many communication to potentially large sets of receivers with different bandwidth requirements. Scalability in RLC comes from three characteristics: each receiver takes congestion control decisions autonomously; traffic filtering is implemented by routers on the multicast distribution tree; and a no-overhead scheme is used to synchronize receivers behind the same bottleneck.

Once defined how to solve problems posed by receiver heterogeneity, we have to find a proper layered organization of data. Across the lowest layer the source of data must distribute essential data, while higher layers provide progressive refinement. In addition, data organization has to be robust against packet losses, that are unavoidable over a connection-less, unreliable network.

Taking into account these two working conditions (i.e. layered organization of data and robustness against packet losses), we have designed and implemented a layered transcoder specific for videoconference transmissions.

We choose to focus on videoconference for two reasons:

1. videoconferencing is a typical application than can take great advantage from multicast transmission in terms of band consumption.
2. the structure of video signal makes easier to organize data in a layered way, since frame-rate and quality of image are two parameters that give an intuitive and easily implementable solution to layered organization;

In the next section, we give a detailed description of the proposed layered video transcoder (`lvt`). In Section III we report on performances of our application, using parameters like image quality, frame rate, robustness and data redundancy. Related work is discussed in Section IV, while concluding remarks are provided in Section V.

II. Design

Rather than designing a specialized layered video encoder as [3], [7], [12], we have followed a different approach, namely developing a stand-alone video transcoder that can be connected to existing videoconference applications, such as VIC [5], to provide easy-to-deploy adaptive and scalable multicast services.

There are four requirements for our video transcoding algorithm. It has to be:

- **layered**, in order to adapt to receivers with different bottleneck bandwidths to the source;
- **cumulative**, i.e. data on higher layers give progressive refinement to data of lower layers. This is not mandatory but grants a better use of available bandwidth, reducing replication of data;
- **low complexity**, so there is no excessive overhead in the transcoding nodes or the receiver;
- **robust** to packet losses that are unavoidable on IP networks.

Defining a layered data organization is essentially a source coding problem, and finding a suitable data arrangement for the various layers depends entirely on the application domain. Ideally, we would like that there is no replication of data sent on the various layers. However this could be contrasting with robustness to packet losses, as the loss of some data (e.g. packets transmitted on the lowest layer) might render other data packets useless for reconstruction purposes.

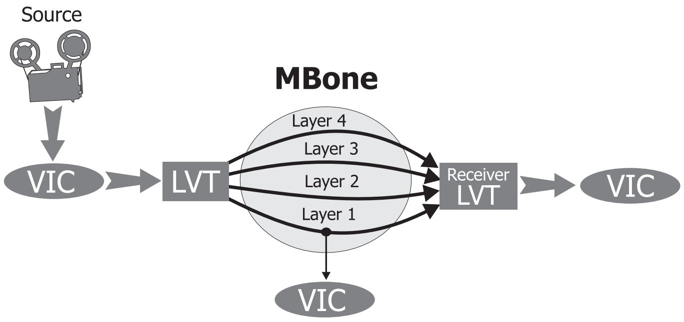

Fig. 1. `lvt` intercepts VIC output packets and distributes them, properly rearranged, over four layers, suitable for transmission on different multicast groups.

In `lvt` we have privileged robustness, and this forced us to tolerate a small amount of inefficiency on bandwidth usage at high bandwidths. We will see from Section III that this inefficiency is small and permits us to achieve better performance in presence of bandwidth limitations.

A. Architecture

As shown in Fig. 1, our transcoder (or better, transcoder pair) acts as a filter between an H.261 video source, such as VIC, and H.261 receivers.

On the source side, `lvt` intercepts data generated by VIC and distributes them over four layers. On the receivers side the transcoder merges received layers into a single data flow, which is then processed by VIC. `lvt` provides on each layer an H.261 compliant video stream, thus a receiver could also receive the first layer of the organization without the need of the transcoder.

One of the video compression algorithms used by VIC is an adaptation of ITU-T H.261 [2], called Intra-H.261. This algorithm inherits the general structure of H.261 but excludes any function based on inter-frame information, like motion vectors. Inter-frame algorithms (e.g. MPEG-2, H.261) use the similarity between adjacent frames to increase the compression ratio. These algorithms require a tight coordination between the source and the receiver: they must share the same information in order to correctly decode differential information. This degree of coordination is not easy to achieve on multicast session, due to the need to collect detailed per receiver loss reports. Therefore inter-frame algorithms are not a good choice for multicast video transmissions over the Internet.

Intra-H.261 uses then *conditional replenishment* to increase compression ratio avoiding this way the use of differential information. Same as H.261, Intra-H.261 splits the image in macroblocks — composed of four blocks of 8x8 pixel — and treats macroblocks separately from each other. Then, according to conditional replenishment, each macroblock is compared with the previously transmitted version of the same macroblock. If differences between the two macroblocks are small (i.e. below a given threshold) then the new macroblock is not transmitted at all and discarded. In addition, to compensate losses that might affect macroblocks that rarely change, a background replenishment process periodically scans the whole frame transmitting "idle" macroblocks.

Conditional replenishment guarantees a low complexity of compression and a good compression rate, because only blocks that exceed the threshold are encoded; and, especially, it is robust against packet losses since each block contains all information needed for its decoding.

These advantages are gained at expenses of quality of video signal. There will certainly be inconsistencies in the image received, and receivers can lose small motion artifacts. But, since our focus is on videoconference transmission, we can release a lot of constrains about quality of video signal.

B. Layered organization

Once defined the compression standard we have to choose information for each layer. As a design goal, higher layers should give progressive refinement to video signal transmitted on lower layers. Progressive refinement means a better quality of image and a higher frame rate: these are the only parameters we can operate on.

Therefore we can describe layered organization of data on a two dimensional space where horizontal axis represents frame-rate and vertical axis represents quality of image, as done in [7]. Layering data means tracing a trajectory over this two dimensional space and then move along it to modify quality of video signal and bandwidth requirements, or, in other words, to change level of subscription (Fig. 2).

At this point, the problem is to choose a good trajectory for our purposes. We have evaluated several possible trajectories using three qualitative parameters:

- quality of video. This is obviously the most important parameter for evaluating efficiency of an organization of data. A good organization must provide a gradually improving quality of video from layer to layer. However, defining quality of video signal is not a trivial problem, especially if using a compression scheme like conditional replenishment. Just using the signal-to-noise ratio of the received image is certainly unsatisfactory, since the frame-rate is a fundamental parameter for quality of video. Frame rate gives video motion feel, and it is hard to define when it is too low, because it depends strongly on the subject. In the design of `lvt` we have decided to privilege completeness of the video rather than high frame-rates, since we believe this gives us best performance in terms of overall video quality.
- bandwidth control policy. The available bandwidth for each layer must be defined, and mechanisms must be introduced to control and possibly limit the transmit rate of a layer. These mechanisms have to reach a trade off between the need for controlling bandwidth utilization and the need of obtaining an acceptable quality for the video signal. We experimented with several policies like discard more blocks, increase compression rate for each block, discard successive frame in order to transmit entirely the one under evaluation, discard the entire frame. A good trajectory is one that allows to use all these policies, letting application free to choose which one is better each time it is required to limit transmit rate.
- overall complexity. If the overhead given by the layered organization of data is excessive, it may be possible that our application does not succeed in process-

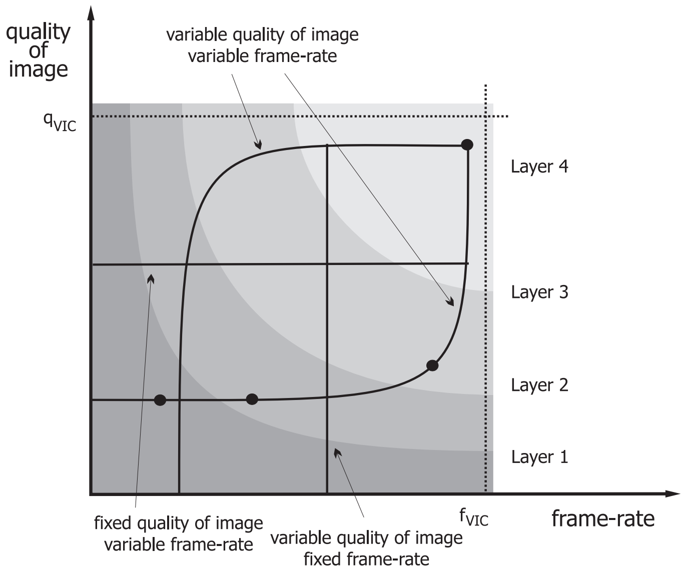

Fig. 2. Four possible layered video organizations: fixed quality, suitable for video with large still background and small high motion artifacts; fixed frame-rate, suitable for slides and still images with slow motion; and two different ones with variable quality and frame rate.

ing all data received from the source, reducing this way the maximum frame-rate achieved. However, this parameter is very difficult to evaluate — it strongly depends on implementation — and only a qualitative estimate can be given.

B.1 Trajectory of `lvt`

In Fig. 3a we show the trajectory that we chose for `lvt`, as this is the one that in our tests gave the best results according to the parameters mentioned earlier for videoconference-like signals. At the lowest layer receivers have a video with a frame-rate and a quality of image about one-quarter of the one generated by VIC. The second layer doubles frame-rate, leaving quality of image substantially unchanged. The third layer doubles again frame-rate reaching this way the frame-rate generated by VIC, while the fourth and highest layer gives refinement for quality of image in order to obtain a video of quality comparable with the output of VIC.

In Fig. 3b we show the organization of frames through the four layers. The label L means that the frame may be a reduced quality version of the one coming from the source. In that case, our layered coder has modified quality of image increasing compression rate of some of the macroblocks of the frame. The task of the fourth layer is to transmit data lost in the lower layers.

This trajectory grants an acceptable quality of video, even at the lowest layer, thanks to the low frame-rate imposed. In addition, frame-rate is doubled from layer to layer reaching nominally the maximum achievable — i.e. the frame-rate of VIC — at the third layer.

The computational complexity for the trajectory shown is very low at the receivers, and only moderately high at the transmit transcoder. The latter has in fact to compute the low quality frame (marked with L in Fig. 3b), and that means a decoding/encoding computation. On the other hand, receiving transcoders only need to merge, filter and forward received packets to VIC without any decoding involved.

B.2 Bandwidth control policy

According to Fig. 3, in principle the first layer processes only one fourth the frames transmitted by VIC, and is designed to use 1/8 of the nominal source bit rate. The other layers use a similar scheme, only working on different frames and target bit rate (respectively 1/8, 1/4 and 1/2 of the nominal source rate).

In practice, due to the H.261 compression and conditional replenishment, the size of each frame is highly variable, so there are no means we can assert that this layer can transmit all frames it receives, and in some cases we need to drop received data because of lack of bandwidth. Moreover, the first layer is missing information of the three intermediate frames (which are normally transmitted on other layers). Because some blocks are not retransmitted over all frames (due to conditional replenishment), receivers subscribing only to the lowest layer might miss updates to these blocks.

These problems – which would cause inconsistent video – are common to the first three layers, since the second layer is missing information sent on the third one, while the third layer is missing refinement information transmitted on the fourth layer.

In order to decide which blocks to send, `lvt` thus keeps a copy of the most recent frame, updated with all blocks transmitted by the source. For each block we also store some additional information, namely the list of layers that have transmitted this block, and the quality used to transmit the block (to save bandwidth, `lvt` can decide to transmit a block using a lower quality than the one used by the source). These additional information are used to decide whether a block has to be transmitted on a given layer,

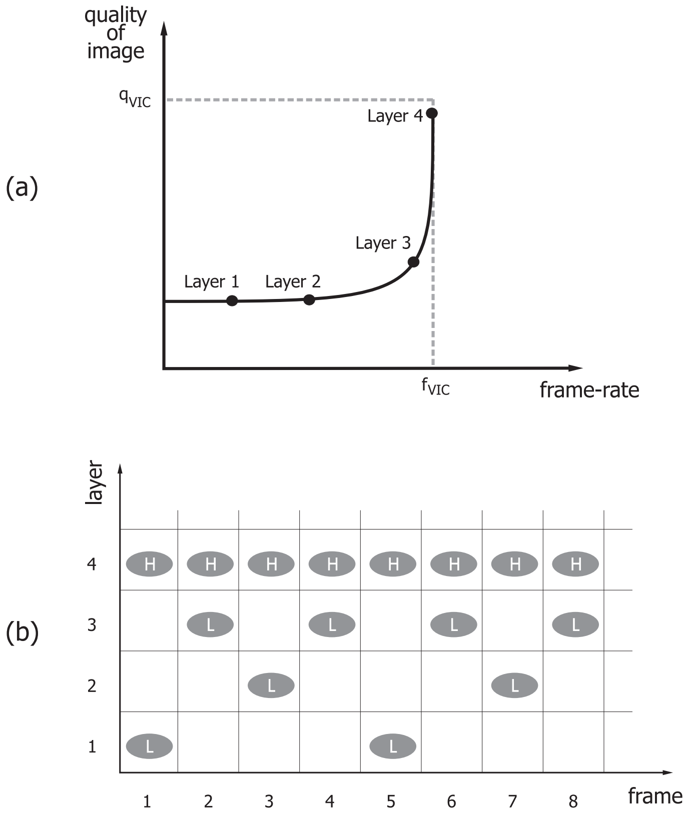

Fig. 3. (a) The trajectory we chose to use in `lvt`. (b) frames normally used on the various layers. L means that the block can be recompressed by `lvt` using a coarser quantization, while H means the block is used as received from the source.

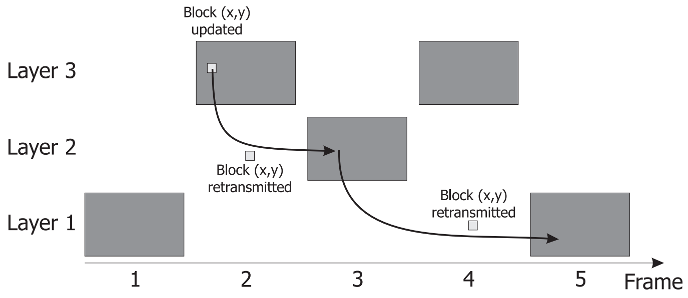

Fig. 4. `lvt` records block updates and can retransmit a block on multiple layers to help keeping the image consistent for receivers at all subscription levels.

even if it was received on a frame that does not directly belong to that layer.

Consider for example the situation shown in Fig. 4. Frame 2 is scheduled for transmission on layer 3, which transmits the update of the block (x,y). Frame 3, scheduled for transmission on layer 2, does not include block (x,y) since it has not changed from frame 2. Without any dedicated mechanism, receivers that are not subscribed to layer 3 will miss the update of block (x,y), and will experience an inconsistent image. Instead, we mark block (x,y) in our frame buffer as "transmitted only on layer 3", so that layer 2 becomes aware of the possible inconsistency and, therefore, will retransmit the block.

The main consequence of this replication of data is that the compression rate of this layered organization of data is slightly worse than the one provided by the source. As a side effect, bandwidth requirements for transmitting frames along each layer may exceed the band we assigned them. Therefore, we introduce a bandwidth control policy as follows:

- if all the macroblocks of the frame to be transmitted exceed the available bandwidth, the coder compresses low-priority macroblocks by using a coarser quantizer;
- the priority of a macroblock is computed each time that macroblock is updated by the source, and is calculated as the distance — by luminance factor — between the new and the old macroblock;
- the coder compresses as many blocks as required to reach the target data rate. If this condition cannot be fulfilled the frame is discarded and the credit is accumulated for the next frame.

The fourth layer has to refine data transmitted on lower layers. It retransmits all blocks that have been compressed, or transmits frames that has been entirely discarded. Two are the policies used for bandwidth control over this layer, depending on data to be sent: i) if an entire frame has to be sent, this layer uses exactly the same policy of the other layers, so it is possible that a frame is completely discarded; ii) if the fourth layer is sending only hi-quality blocks, it sends them as long as it has available bandwidth.

C. *Receiver side*

At the receiver side, we develop a light-weight process (Receiver-`lvt`) that just performs some data rearrangement before sending data to VIC. This process has to:

- define the subscription level to use (this can be set statically or changed dynamically e.g. under control of a congestion control scheme);
- integrate data received from the set of multicast groups in a single stream to transmit to VIC;
- performs some control over possible data missing or misplaced.

Receiver-`lvt` needs three information to correctly perform the operations described above: the sequence number of the packet to recognize the correct sequence of packets and to discard duplicates; the timestamp of the frame contained in the packet; the indication of end of frame. These information are already present in the Real Time Transport Protocol (RTP) header [10] which is used to encapsulate frames generated by the source. Information specific to H.261, i.e. the H.261 description of each block of the image (including coordinates), are positioned after the RTP header [11].

The last problem the decoder has to cope with is the recovery of original delay between frames. Since most of the work is performed directly by VIC, we need only to merge the four layer data stream in a single stream in the correct order to avoid wastes of data. Therefore we implement a frame buffer that simply stores information received by each layer.

D. *Inter-operability*

The transcoder has been designed with the main purpose of providing a useful tool for evaluating performance of multicast layered congestion control protocols. We chose to avoid imposing any constrain to the videoconference application in order to preserve the maximum rate of flexibility. Indeed `lvt` imposes some requirements only on video input stream. The video stream has to be H.261 and RTP compliant, even if in principle can be adapted to any block-based encoding. Our application does not require the use of conditional replenishment for video compression. In addition, the integration between `lvt` and *RLC* is straightforward, since `lvt` does not make any assumption on protocol behaviour.

This type of inter-operability has been achieved at the expense of some performance of our layered video encoding. Indeed the choice of not making changes to VIC source code, and even not making any assumption on the computations performed by VIC or by *RLC*, allows us to provide an highly adaptable tool for layered video encoding.

III. Performance

Definition of performance of a layered video coding is not trivial. The most useful parameters for defining the overall quality of a layered video coding are the following:

- **quality of image**, usually defined as signal-to-noise ratio between the compressed image and the original image. For simplicity, in order to evaluate the quality of image, we compute the number of macroblocks updated per frame and the mean H.261 quantizer of the image. We believe that these parameters are meaningful for evaluating the consistency between the received and the original video.
- **frame-rate**, defined as the number of frames per second transmitted, is a measure of the smoothness of video. This parameter, alone, is certainly not optimal for computing performance of a video coding since in a limited bandwidth environment it strongly depends on image quality. However, in our layered compression scheme, this parameter can be meaningfully used because, as shown in the previous section, we force our layered coder to always send frames with all blocks received by VIC.
- **robustness**, defined as the resilience of our coding to packet losses. This is a parameter very difficult to compute and we base our analysis on evaluation of quality of image and frame-rate in a highly lossy environment.
- **redundancy**, defined as the number of macroblocks that are duplicates of macroblocks already sent on higher layers.

An intuitive way to interpret `lvt` operation is as a selective packet dropper. In this view, we can evaluate its performance by comparing the behavior of a source S with bandwidth B through a bottleneck link of capacity $B' < B$, with that of a subset of layers generated by `lvt` – when driven by the same source S – with an aggregate bandwidth matching the bottleneck.

In order to achieve a qualitative, in addition to a quantitative, performance evaluation, we run our experiments using the setup of Fig. 5. Actual source (a playback of a previously recorded VIC stream), sinks (VIC receivers) and transcoder (`lvt`) were used, together with separate programs to extract statistics from the received data stream.

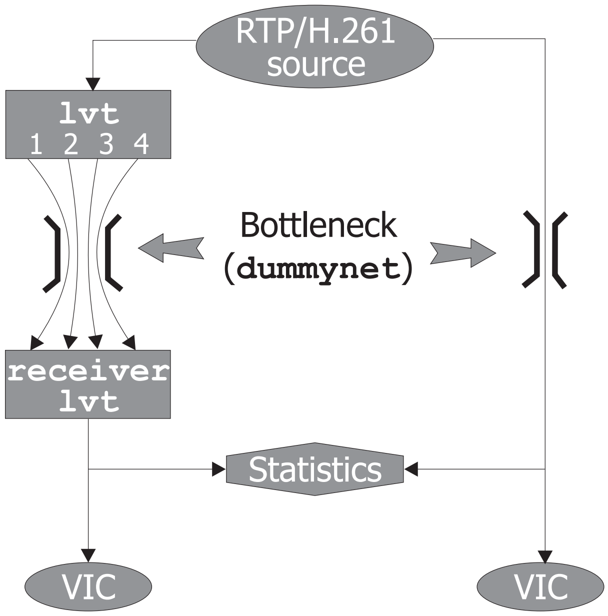

Fig. 5. The setup used for our experiments.

The bottlenecks were simulated using `dummynet` [8], a flexible traffic shaper/link emulator developed by one of the authors and part of FreeBSD, which permits to experiments with different capacity, queue size, delays and losses.

The graphs shown in this section refer to a full VIC source — a typical conference talk — recording at 256 Kbps, CIF resolution (352x288) and a peak frame rate of 30 fps. For each bottleneck bandwidth (32, 64, 128 and 256 Kbps), we then compare macroblock update rate, mean quantizer and frame rate between S and `lvt` session with an aggregate bandwidth equal to the bottleneck capacity. In the experiments we uses a queue size for bottleneck links of 40 packets; we could not use shorter queues because VIC is extremely bursty, and relies on a packet with marker bit set to display a frame. Because this packet is the last one in a burst, shorter queues would negatively affect measurements without the transcoder.

In the following sections we call:

- S_{256} the source encoded at 256 Kbps;
- B_{32}, B_{64}, B_{128} and B_{256} the source S_{256} through a bottleneck link of capacity 32, 64, 128 and 256 Kbps;
- L_{32}, L_{64}, L_{128} and L_{256} the output generated by `lvt` on the first, second, third and fourth subscription level when driven by S_{256}.

All measurements were done on a Pentium Pro 200 MHz. Given the relatively low data rate used in the experiments, the CPU speed does not limit performance.

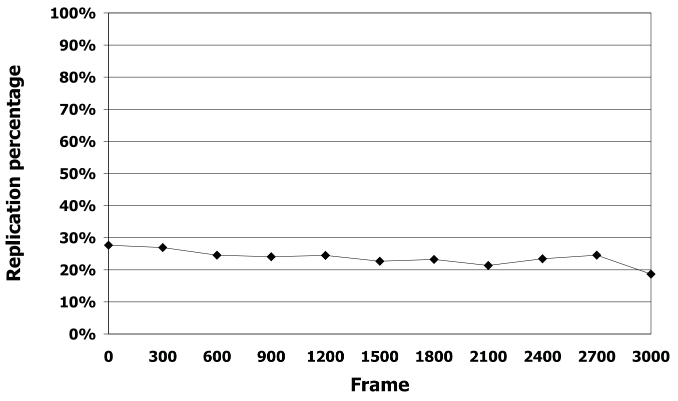

Fig. 6. Ratio between the number of H.261 macroblocks that have been already sent over lower layers, and the total number of macroblocks sent per frame.

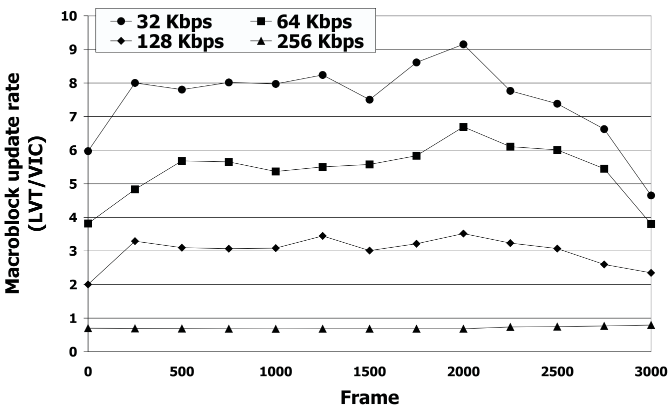

Fig. 7. Ratio between `lvt` macroblock update rate and VIC macroblock update rate.

A. Data redundancy

The first parameter we present is the amount of data redundancy generated by `lvt`, because this parameter gives a key for interpreting other results.

As mentioned in Section II-B.2 some redundancy is useful in a layered organization to let receivers tolerate losses and change their subscription level without experiencing inconsistencies in the video received. The amount of redundancy is dependent on the algorithm used to select blocks to transmit at each layer, as described in Section II-B.2. In the current implementation, `lvt` does not do any filtering, and a block transmitted on the highest layer can in principle be replicated on all four layers. Of course, the actual amount of replication depends on the source characteristics.

The average ratio between the total number of macroblocks sent and the number of macroblocks that have been already sent over lower layers is about 25% (Fig. 6). In detail (not shown on the graph), replicated data used the 58% of the available bandwidth of the first layer, the 48% of the second layer bandwidth and the 29% of the third layer.

B. Quality of image

As discussed above, we base the analysis of quality of image on the macroblock update rate, i.e. the number of new macroblocks visualized per frame. This parameter gives an evaluation of the consistency of the image on receivers with the original image.

In Fig. 7 we show the ratio between the `lvt` macroblock update rate and VIC macroblock update rate for the pairs L_{32}/B_{32}, L_{64}/B_{64}, L_{128}/B_{128} and L_{256}/B_{256}. As we see from the graphs, `lvt` outperforms the source in all cases where the bottleneck bandwidth is lower than the source bandwidth. This was well expected, as the selective discard process performed by `lvt` is aware of the bottleneck bandwidth. On the other hand, on a 256 Kbps link `lvt` grants about 70% the VIC macroblock update rate. This was also expected and perfectly in line with the amount of redundancy introduced by `lvt` and computed in the previous section.

`lvt` can also act on the quantizer of H.261 blocks to save bandwidth. In Fig. 8 we show the ratio between the mean H.261 quantizer used by S_{256} and L_{32}, ..., L_{256}. In this experiment, the ratio very close to 1, meaning a somewhat comparable resolution of the image. Two factors influence this result: i) we compute the mean on all the 396 macroblocks that make up a frame, even if a small set of low quality macroblocks can reduce the overall "visual sense"; ii) according to `lvt` bandwidth control policy, lower layers can discard an entire frame saving the bandwidth for next frame that can be then sent at a good quality.

C. Frame rate

In Fig. 9 we show the ratio in frame rate achieved between $B_{32} \ldots B_{256}$ and $L_{32} \ldots L_{256}$ respectively. These results are very similar to those obtained for quality of image. Indeed `lvt` outperforms VIC over every bottleneck while grants a frame rate of about 75% that of VIC over a links with enough bandwidth to avoid packet losses.

L_{256} performs about 25% worse of B_{256} due to high degree of frame discards on lower layers that cannot be entirely recovered by the fourth layer. These frame discards are a consequence of the high redundancy of data on lower layers.

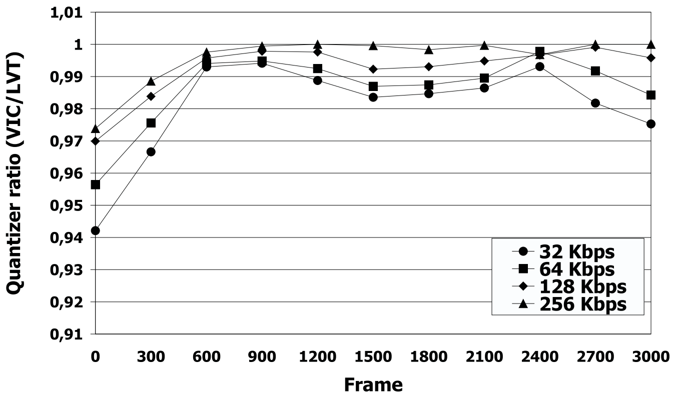

Fig. 8. Ratio between $S_{25a}6$ and L_{32}, ..., L_{256} mean H.261 quantizer.

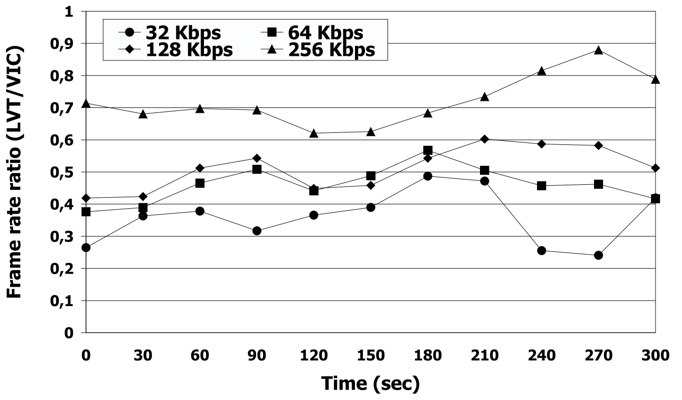

Fig. 10. Ratio between L_{32}, L_{64}, L_{128} and S_{32}, S_{64}, S_{128} (i.e. source video S encoded at 32, 64 and 128 Kbps, respectively).

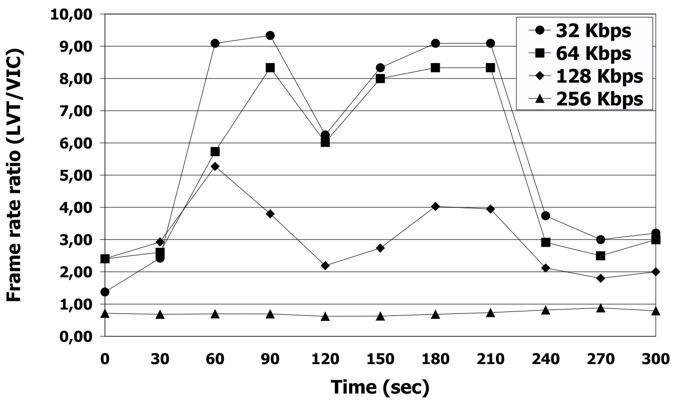

Fig. 9. Ratio between `lvt` and VIC frame rate rate for the 256 Kbps stream through bottleneck links.

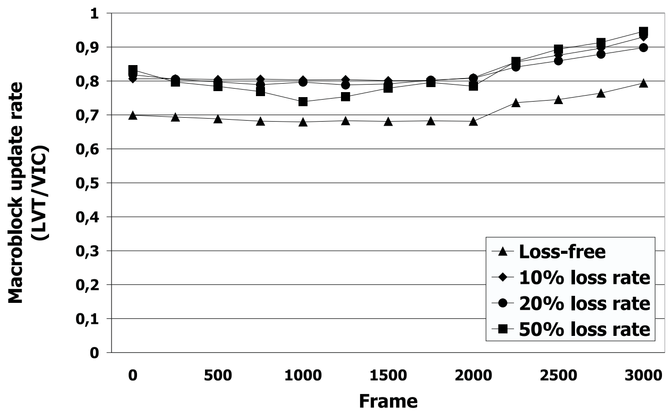

Fig. 11. `lvt` versus VIC behaviour at 256 Kbit/s in presence of losses – macroblock update rate.

We have also compared the frame rate achieved with $L_{32} \dots L_{256}$. and $S_{32} \dots S_{256}$ (i.e. source video S encoded at $32 \dots 256$ Kbps, respectively), and the results are shown in Fig. 10. Compared to a source coded at a comparable rate, `lvt` always yields a lower frame rate, ranging from about 0.75 times for the highest rate, to an average of 0.4-0.5 times at the lowest rates. This is partly due to the amount of redundancy which is present in data generated by `lvt`, and partly to the relatively simple strategy used by `lvt` in selecting which blocks to transmit. This result indicates an area of the transcoding algorithm that might benefit from further investigation.

D. Robustness

In order to evaluate the robustness of our layered video transcoder we compare S_{256} and L_{256} over a link with a packet loss rate of 10%, 20% and 50%. As usual, we used `dummynet` for configuring bottleneck characteristics.

In Fig. 11 we show the ratio between L_{256} and S_{256} macroblock update rate. In presence of losses `lvt` shows a modest improvement compared to VIC. This was expectable due to the redundancy in data transmitted by `lvt`. In Fig. 12 we show the ratio between `lvt` and VIC frame rate, again under various loss conditions. In this case VIC and `lvt` performs in the same way because the frame rate corresponds to the frequency of packets with the marker bit set, and these are not replicated by `lvt`. Overall, these experiments show that `lvt` provides video coding at least as robust as the source.

IV. Related work

Layered multicast data distribution and layered video coding are relatively recent research topics. The need for a layered coding, where receivers can dynamically change

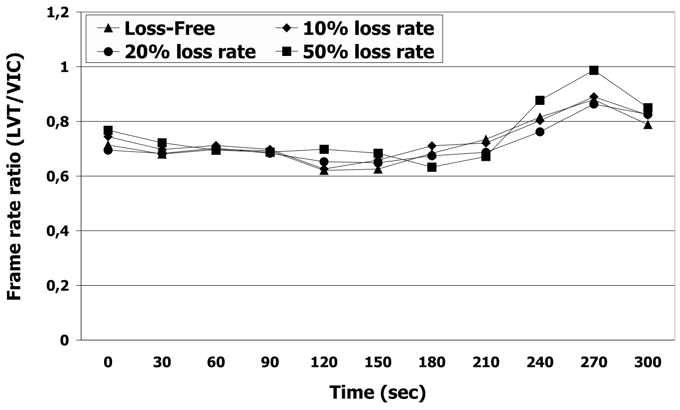

Fig. 12. `lvt` versus VIC behaviour at 256 Kbit/s in presence of losses – frame rate.

their subscription level, comes from some recent work about multicast congestion control schemes, as [4], [6], [14], which permit receivers to select the number of layers they subscribe to by looking at congestion signals using hierarchical [4] or fully distributed [6], [14] protocols. The design of layered video encoders is presented instead in [3], [7], [12], [13], [15].

The encoder proposed in [7] uses block-based conditional replenishment as defined in Intra-H.261 [5], and combines progressive DCT and subband decomposition to obtain various qualities of image at each layer. In addition, this encoder provides a frame organization over different layer similar to the one shown in Fig. 3b.

The encoder presented in [3] is an adaptation of conventional predictive encoders with a special buffer to store multiple frame references. This solution is required to compensate the loss of the reference frame when receivers hop between layers.

Vickers et al. [15] layered encoder requires additional support from the network, such as an explicit congestion notification or a hop-by-hop flow control. This way, the encoder can enhance the overall bandwidth utilization by dynamically adjusting the number of video layers or the rate of each layer.

Other schemes propose the use of forward error correction or retransmission [4], [12]. These models provides a different approach for increasing compression performance: they increase the reliability of transmission in order to make use of less resilient video coding algorithm that grant a better video quality.

V. Conclusions and future work

In this paper we have presented a layered video transcoder, `lvt`, suitable for the multicast transmission of video signal to heterogeneous receivers. The work was mainly motivated by the need of providing layered video sources to be used in conjunction with layered multicast distribution schemes such as RLC [14]. We focused our work on a transcoder approach, rather than designing a layered encoder, because this provides an easy way to deploy scalable multicast services using existing video sources or recordings. Further, we focused on low complexity transcoding schemes so that low overhead transcoding services could be easily located either at the source or within the network.

Our design goals forced as to partially sacrifice performance at high data rates (e.g. in terms of frame rate) in favour of simplicity and robustness. Nevertheless, the performance analysis presented in this paper shows that `lvt` performs reasonably well at the topmost data rate, and significantly outperforms the source in presence of bandwidth limitations, permitting an effective adaptation of the source to the bottleneck data rate.

While the actual implementation of `lvt` focus on an RTP/H.261 stream, the approach we followed is equally applicable to other block based video encodings.

Future work will focus on more effective techniques for selecting which blocks to transmit, thus removing part of the data replication that exists in the current implementation. We also plan a more comprehensive performance analysis when the algorithm is used in conjunction with a layered congestion control scheme and thus receivers do relatively frequent hopping between different layers.

References

[1] W. Fenner, *Internet Group Management Protocol, Version 2*, INTERNET-DRAFT (working draft), January 20, 1997, `http://ds.internic.net/internet-drafts/-draft-ietf-idmr-igmp-v2-06.txt`

[2] ITU-T, *Video codec for audiovisual services at p*64kb/s*, International Telecommunication Union Recommendation H.261, 1993.

[3] Jae-Yong Lee, Tae-Hyum Kim and Sung-Jea Ko, *Motion Prediction Based on Temporal Layering for Layered Video Coding*, Proceedings of ITC-CSCC '98, Vol. 1, July 1998.

[4] X. Li, S. Paul, P. Pancha and M. Ammar, *Layered Video Multicast with Retransmissions (LVMR): Evaluation of Hierarchical Rating Control*, Proceedings of INFOCOM '98, March 1998.

[5] S. McCanne and V. Jacobson, *vic: a flexible framework for packet video*, Proceedings of ACM Multimedia '95, November 1995.

[6] S. McCanne, M. Vetterli and V. Jacobson, *Receiver-driven Layered Multicast*, ACM SIGCOMM '96, August 1996.

[7] S. McCanne, M. Vetterli and V. Jacobson, *Low-complexity Video Coding for Receiver-driven Layered Multicast*, IEEE Journal on Selected Areas in Communications, vol. 16, no. 6, pp. 983-1001, August 1997.

[8] L. Rizzo, *Dummynet: a simple approach to the evaluation of network protocols*, ACM Computer Communication Review, Vol.27, n.1, pp.31-41, January 1997.

[9] L.Rizzo, L.Vicisano, *RMDP: an FEC-based Reliable Multicast*

protocol for wireless environments, ACM Mobile Computing and Communications Review, Vol.2, n.2, pp.23-31, April 1998.

[10] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, IETF, Audio-Video Transport Working Group, January 1996, RFC 1889.

[11] H. Schulzrinne, *RTP Profile for Audio and Video Conference with Minimal Control*, IETF, Audio-Video Transport Working Group, January 1996, RFC 1890.

[12] Wai-tian Tan, Avideh Zakhor, *Multicast Transmission of Scalable Video using Receiver-driven Hierarchical FEC*, Packet Video Workshop '99, April 1999.

[13] D. Taubman and A. Zakhor, *Multi-rate 3D subband coding of video*, IEEE Transaction on Image Processing, vol. 3, no. 5, pp 572-588, September 1994.

[14] L. Vicisano, L. Rizzo and J. Crowcroft, *TCP-like congestion control for layered multicast data transfer* IEEE INFOCOM '98, March 1998.

[15] B. Vickers, C. Albuquerque and T. Suda, *Adaptive Multicast of Multi-layered Video: Rate-based and Credit-based Approaches*, Proceedings of INFOCOM '98, pp. 1073-1083, March 1998.